Deduplication-based Energy Effcient Storage System in Cloud Environment

Deduplication-based Energy Efficient Storage System in Cloud Environment

He Li[1], Mianxiong Dong[2], Xiaofei Liao[1] and Hai Jin[1,*]

[1]*Services Computing Technology and System Lab, Cluster and Grid Computing Lab*
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
[2]*National Institute of Information and Communications Technology*
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289 Japan
[*]*Corresponding author: hjin@hust.edu.cn*

In the cloud computing, companies usually using high-end storage systems to guarantee the efficiency of virtual machines (VM). These storage systems cost a lot of energy for their high performance. In this paper, we propose EEDS, a deduplication-based energy efficiency storage system for VM storage. We firstly investigate some VM image files with general operation systems. With the analysis result, we find there are many redundant data which bring extra energy cost in VM storage. Therefore, in EEDS, we design a two-step deduplication mechanism to reduce these redundant data without service interruption while traditional deduplication technology is used for offline backup. Since this mechanism needs CPU time which is the limited resource, we design a deduplication selection algorithm such that the storage energy consumption is minimized for a given set of VM in a cloud cluster with limited resource for deduplication. Experiment results in the para-virtualization environments with EEDS show that the energy consumption is reduced by even up to 66% with negligible performance degradation.

Keywords: Energy Efficiency; Green Computing; Cloud; Virtualization; Deduplication

1. INTRODUCTION

In cloud data center, the virtualization technology brings better scalability and resource utilization. Meanwhile, since the limitation of the I/O virtualization technology, the ordinary hardware is hard to provide enough performance for virtual machines (VM). To meet the requirement of fast I/O support, cloud service providers usually adopt high-end storage systems for VM storage[1]. These high-end systems usually means high energy consumption[2, 3, 4].

To reduce the energy consumption of storage systems, many works focus on the scheduling of storage systems to make sure that more hard disk drives drop in sleeping status. Scheduling is effective to keep the energy consumption down with varying loads in which low load bring less storage energy consumption. Therefore, considerring scheduling is hard to reduce more energy consumption by a given storage load, we choose another way that cutting down the existed storage loads in cloud environment.

Reducing the storage load is another way to decrease the energy consumption of the storage system[5, 6]. Usually, for better isolation and management, VM storage systems encapsulate the entire data of each VM into a virtual disk image file. While there are limited types of operation systems and softwares in a cloud data center, many similar data blocks are existed between VM image files. Except that some images extended from the same original version, it is hard to shared data between VM images. Therefore, we consider these data as the one type of redundancy which bring extra loads in VM storege.

Another redundant data are existed in each disk image file. Image files are regarded as normal large files. Each image file can be divided to many small blocks. In these blocks from same image file, it is not difficult to find identical blocks. These blocks can be considered as the redundant data. To find these two types of redundant data, we firstly investigate the redundant data by analyzing some VM image files with different OS installed.

Based on the analysis result, we design and implement EEDS, a storage system for reducing the energy consumption by eliminating the redundant data during VM storage procedures. In EEDS, we

adopt some similar mechanisms from the deduplication technology. Deduplication is a common technology in data backup with high data compression ratio. Unlike the traditional deduplication which needs service interruption, the deduplication mechanism in EEDS, which named online deduplication, is a real-time processing and agnostic to all guest VMs.

Another problem is, since removing redundant data needs extra time of CPU processing, it is possible to influence the quality of service (QoS). Considering the processing resource in a cloud data center is limited, it is hard to deduplicate each image files in the cloud environment. Therefore, we module this problem as an deduplication selection problem to choose which image files needs deduplication. We also proof this problem is a NP complete problem. In EEDS, we use a dynamic programming algorithm to find the lowest energy consumption with a given work loads.

In this paper, we also describe our implementation of the demonstration system based distributed storage systems with Xen Cloud Platform(XCP) [7]. We insert data calculation process module in the user layer for rapid development. Therefore, we evaluated the online-deduplication both on this distributed environment and a single plane environment.

The main contributions of this paper are summarized as follows.

- First, we design a new mechanism named online-deduplication for redundant eliminaiton, which reduces the work loads in VM storage for better energy consumption.
- Second, we design EEDS, a storage system to support online-deduplication in a cloud center platform. We implement our design and evaluate the EEDS system in a small cloud cluster.
- Third, we study the deduplication selection problem to minimize the energy consumption with limited the deduplication process resource.

The rest of this paper is summarised as follows. We discuss the related works in Section 7. The methodology of online-deduplication is discussed in Section 2. We model the deduplication selection problem and describe the heuristic algorithm in Section 4. In Section 6, we evaluate the demonstration of our design to verify our methodology.

2. MOTIVATION

In this section, we firstly analysis some existed VM image files to character the redundant data in VM storage. We also introduce the traditional data deduplication technology in data backup.

2.1. VM image redundancy

We investigate VM image files to find the redundant data in VM storage. We analyze five image files from a desktop virtualization system named ClouDesk[8]. This image files have been installed on five different OS including Fedora 15, Fedora 16, CentOS6, Windows 7 and Windows 8.

TABLE 1. The redundancy of virtual machine image file

	Fedora15	Fedora16	CentOS6	Win7	Win8
Fedora15	34.07%	50.94%	32.97%	52.48%	28.91%
Fedora16		31.44%	31.57%	50.12%	28.02%
CentOS6			31.65%	52.39%	27.63%
Win7				63.71%	47.05%
Win8					24.69%

Different from inter-redundancy, inner-redundancy is a type of redundancy come from single disk image file. Since the disk image files are very big file whose length will reach several to dozens of gigabyte, some data in single file will repeat many times. For a direct impression, we tested some image files to find the inner-redundancy. We scan three image files to present the most common scene in a virtualization environment. This Scan is like the first step in file pressuring but much simpler that only reflects the data repeatedly. As shown in Table 1, we list the redundancy in three image files after scanning and summary and the inner-redundancy of each image are the data above the diagonal. This type of redundancy cannot be removed by any COW strategy since its complicate distribution in the image file.

2.2. Data Deduplication

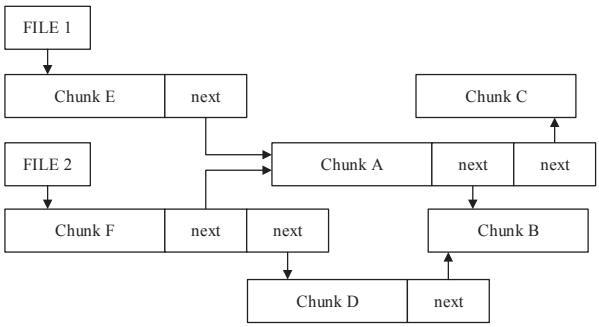

FIGURE 1. Example of data deduplication in tranditional backup systems.

Deduplication is an important technology for backup in large scaled data center, which eliminate redundancy data by a specialized data compression technology. We discuss an example of an typical data deduplication method for better understanding. In this example, a data chunk consists of three parts, the data stored in this chunk and some points to the successor chunks. There are two files, FILE 1 and FILE 2, need deduplication for redudancy elimination. In FILE 1, there 4 chunks which are Chunk E, A, B and C. To FILE 2, the sequance of chunks is FABCDB. Without deduplication, we need to use 10 chunks to store these two files. Considerring there are only 6 different chunks,

we use some points to link these unique chunks together in this example as shown as FIGURE 1.

In these linked chunks, to access FILE 1, we can get the entrance to Chunk E. Chunk E has a point to the chunk A. In chunk A, there two points: first one is point to Chunk B and the second one is point Chunk C. If a chunk has no point, the access sequence is back to the previous Chunk and find the next point. Therefore, since Chunk B has no point, the access sequence is back to the next point linked to Chunk C. To FILE 2, there two chunk with two points: Chunk F and Chunk A. The first point in Chunk F is linked to the Chunk A which is the as same as FILE 1, and the second point is point to Chunk D which has only one point linked Chunk B. With these link relationship, we can use 6 chunks to store 10 chunks, which means there 4 chunks are eliminated.

However, since the structure could be designed to reduce the difficulty of realization of deduplication like traditional chain structure, it brings a serious problems to the storage performance that the access latency to chunks is increased with these points. From the example, when an application want to access some data in Chunk C of FILE 1, it needs three jumps during the addressing, which means additional three reading to the storage system. In data back up, this latency is not a problem while the history data are usually cold data without frequently access while in VM storage, it will bring terrible performance degradation. As a result, in EEDS, we propose an new deduplication mechanism with an directly mapping to each chunk with a constant latency. We describe this mechanism in the next section.

3. DESIGN

In this section, firstly, we describe the system structure to support online deduplication in EEDS. Then, we discuss the detail of the meta data of VM images for online deduplication.

3.1. System Structure

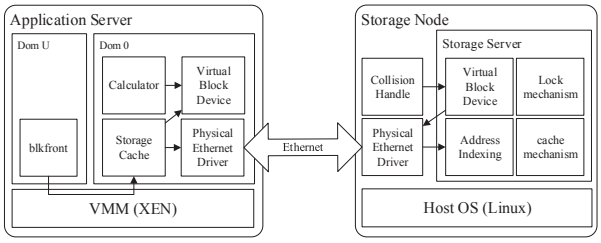

FIGURE 2. Overview of the EEDS architecture.

To treat the redundancy in VM storage for lower energy consumption, we design a new structure of VM storage as shown as FIGURE 2, which consists of three modules: hash value calculator, address index and collisions handler. The hash value calculator generates the hash value of each data chunk by MD5 [9]. The address index has two mapping: one is used between hash values and data chunk addresses, another is between VM images and the chunk addresses. The collision handler checks the newly generated data chunk and the existed data chunk with same hash value to find collision and arrange storage space to store the collision data chunk.

We use MD5 as the fingerprint of data since its low collision frequency even the computation cost cannot be neglected. For better performance, we added a calculate machine used for the hash value generation specially. We package RDP call to process the hash calculator request then feedback the MD5 value. With this fingerprint, after storing data chunk to the storage system, address index will add a mapping between its fingerprint and the physical address. Meanwhile, for accessing the each data chunk efficiently, the address index maintains the index structures between physical addresses and the virtual addresses in all VMs.

The main function of collision handler is checking the hash collision between new data and existed data. Since the address index maintain a mapping between each data block and the fingerprint, when a new data with the same MD5 as before, collision handler checks the each bits of the new data to find whether the new data is repeated. If a collision happens, the collision handler will apply a new space in the storeage system to store the new data.

With this system structure and processing procedures, online deduplication takes many computing resources for each VM images. Considering most of resources are provided for the cloud services, the resources for energy efficiency are limited that only a part of VM images are processed by online deduplication. In the Section 4, we discuss the selection problem that selecting which VM images for optimization.

3.2. Mapping Structure

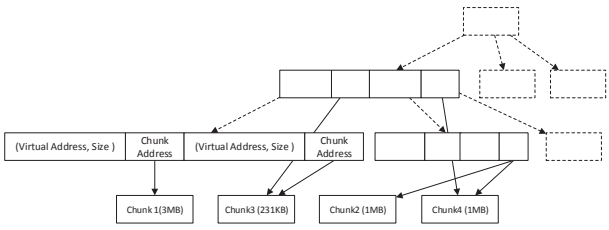

FIGURE 3. The index structure for online deduplication in EEDS.

In particular to minimize the performance sacrifice of online deduplication, EEDS borrows a structure usually used to manage items in the database system. As shown in Fig 3, an address index of each disk image is effectively a single block address space, represented by a B-tree that maps block addresses in virtual disk image

to physical addresses in storage space. Block addresses in each VM are a continuous range from 0 to the size of its virtual disk, while physical addresses reflect the actual location of a chunk on the storage node. In EEDS, the address index maps block address with 4k size or 8k size for leverage the access performance and deduplication efficiency. Mappings are stored in B-trees [10], with less than 6-level depth for rapid reflex, also based on 4K blocks. Radix metadata stores 32 triples and each triple consists of the virtual address, the size of the chunk and the chunk address. We set the maximum chunk as 32MB and use 64bits global physical address then the layout results in a maximum disk image of 512GB.

Another address indexing is between the fingerprint of each data block and its physical address. After a new data was generated, calculator will build two-tuples to store the MD5 value and the count of collision. Unlike the CAS mechanism, since the insufficient relationship between the chunk which is the physical storage unit and the data block, the MD5 hash-value cannot be considered as the address of the corresponding chunk. Because bidirectional mapping is necessary in our mechanism when new data are being inserted, we chose boost-bitmap as the data structure for storing the relationship between MD5 value and physical data block address.

4. PROBLEM STATEMENT

In this section, we discuss the deduplication selection problem that selecting the deduplicating VM images to minimize the energy consumption with limited processing resources.

Firstly, we study the energy consumption in VM storage. We consider four types of energy consumption in our system: cache E_c, storage access E_t, data maintain E_s and deduplication E_d. The consumption of cache means the VMM access local cache instead of storage system access. The storage access energy means the energy consumption during ordinary I/O operation with storage system. The energy used for data maintain means a fix consumption to store data in storage devices. The more data stored in storage system, the more active devices are required. The deduplication energy indicate the consumption of processing duplicated data. Therefore, the total energy consumption of our system is as follows.

$$E = E_c + E_t + E_s + E_d$$

Then, we discuss the scheduling granularity in this problem. In general, access frequency to each data block is different and it reduces more energy consumption by cache the hotspot data. However, to a data block, it is hard to find whether it is a duplicate block after analysis enough data block. Since the energy of comparison processing is the most consumption in deduplicate, analysis the duplicable of each block is not acceptable. We choose VM image as the scheduling granularity since file is easy to predict the access traffic and the deduplication ratio.

After that, we model the energy consumption of each file. We use F to denote the set of all images in storage system. To a image $f \in F$, we use S_f to denote the data size, T_f to denote the predicted access traffic and d_f to denote the deduplicate ratio. In our storage system, the energy to access unit of data in cache is set as e_c, the energy for transferring unit of data is set as e_t, the energy of this fix consumption for maintaining unit of data is set as e_s, and the energy to deduplicate unit of data is set as e_d. Therefore, we set a value x_f to denote whether image f is deduplicated in our system as follows.

$$x_f = \begin{cases} 1 & \text{if } f \textit{ is deduplicated} \\ 0 & \text{if } f \textit{ is not deduplicated} \end{cases}$$

After that, we get the energy consumption E_f to image f as follows.

$$\begin{aligned} E_f &= e_c d_f S_f x_f + e_t(1 - d_f x_f) S_f t_f + \\ &\quad e_s(1 - d_f x_f) S_f + e_d S_f x_f \quad \forall f \in F \\ &= (e_t t_f + e_s) S_f - ((e_t t_f + e_s - e_c) d_f + e_d) S_f x_f \end{aligned} \tag{1}$$

To simplify the equation (1), we use A_f and B_f as follows.

$$A_f = (e_t t_f + e_s) S_f \quad \forall f \in F \tag{2}$$

$$B_f = ((e_t t_f + e_s - e_c) d_f + e_d) S_f \quad \forall f \in F \tag{3}$$

With A_f and B_f, the energy consumption E_f to the image f is described as follows.

$$E_f = A_f - B_f x_f \quad \forall f \in F \tag{4}$$

To simplify the problem, we consider energy is reduced by deduplication that B_f of each image f is nonnegative.

To the whole system, the energy consumption is expressed as follows.

$$E = \sum_{f=1}^{|F|} (A_f - B_f x_f) \quad \forall f \in F \tag{5}$$

In general, since the limitation of hardware performance and the quality or service, the ratio of deduplication processed data is limited by an existed requirement. To model this requirement, we use o_f to denote the overload of deduplication processing to the image f and P denotes the performance limitation of the whole storage system. As a result, the total overload of each file is stated as follows.

$$\sum_{f=1}^{|F|} o_f x_f \le P \quad \forall f \in F \tag{6}$$

And the object of the deduplication selection about each image is or is not deduplicated is find the minimum energy consumption E with the limitation of the formula (6).

THEOREM 4.1. *The deduplication selection problem is NP-hard.*

Proof. We prove the NP-hardness of the deduplication selection by reducing a well-know 0-1 knapsack problem defined as follows.

0-1 knapsack problem. Giving a set items $\{a_1, a_2, ...a_m\}$, each item (a_i) has a value v_i and weight w_i, and a knapsack with maximum weight W, is there a 0-1 knapsack scheme maximum the value of items in the knapsack and their weight is no more than W?

We consider the item f has a value B_f and weight o_f, and a knapsack with maximum weight P. We suppose the deduplication selection problem has a solution that the 0-1 knapsack problem. We use S to denote the items in the knapsack and the sum of value is maximum and the weight is no more than P. In the corresponding solution of deduplication selection problem, we choose S as the same images for deduplication and the total overload in S is less than P.

We than suppose that deduplication selection problem has a solution that the energy consumption is minimum with deduplicated images S. Considering the object that minimum the energy consumption E, in which the $\sum_{f=1}^{|F|} A_f$ is a definite value before the selection. From equation (4), the sum of the B_i in the set S is maximum, which forms a solution of the 0-1 knapsack problem.

It is easy to see that the deduplication selection problem is in NP class as the objective function associated with a given solution can be evaluated in a polynomial time. Thus, we conclude that the deduplication selection problem is NP-hard.

□

5. ALGORITHM DESIGN

In this section, we propose a heuristic algorithm, called DPS (Dynamic Programming for selection), to solve the deduplication selection problem. Its basic idea is using dynamic programming that breaking the selection problem down into simpler subproblems with CPU time slices increased one by one. In each subproblem, we compare each image iteratively to find whether the energy consumption is decreased by add this image. If energy consumption is decreased, we add this image as one existed solution. After computing each possible combination iteratively, we can find the result for the selection problem.

In ordinary operation systems, task schedulers usually use the time slice as the unit for assigning CPU time to each task. To simplify the solution, we make an assumption that processing resource and overload can be described by the number of time slices. Since the time slice numbers are positive integers, we adopt DPS to solute the deduplication selection problem as shown as Algorithm 1.

Procedure 1 The dynamic programming algorithm for deduplication selection problem

```
1: for j from 0 to P do
2:     R_{0,j} ← 0;
3: end for
4: for f' from 1 to |F| do
5:     for j from 0 to P do
6:         if o'_f ≤ j then
7:             R_{f',j} ← max(R_{f'-1,j}, R_{f'-1,j-o'_f} + B'_f)
8:         else
9:             R_{f',j} ← R_{f'-1,j}
10:        end if
11:    end for
12: end for
13: E_min ← Σ_{f=1}^{|F|} A_f − R_{|F|,P}
14: p = R_{|F|,P}
15: D ← ∅
16: for f' from 1 to |F| − 1 do
17:    if R_{f',p} ≠ R_{f'+1,p} then
18:        D ← D ⋃ f'
19:        p ← p − o'_f
20:    end if
21: end for
```

Firstly, we define $R_{f',j}$ to denote the maximum $\sum_{f=1}^{f'} B_f x_f$ that can be attained with overload less than or equal to j deduplicating images up to f'. In line 1 to line 3, we set the initial value of each $R_{f',j}$ is 0. In line 4 to line 12, we use a nested loop to traverse each pair of f' and j. In this loop, we calculate each $R_{f',j}$ that $\forall f' \in F$ and $0 \leq j \leq P$ by adopting the general dynamic programming solution of 0-1 knapsack problem. [11] and the minimum energy consumption is calculated in line 13. After that, we use a set D to find which image files are chosen for deduplication and define p as the processing resource appeared in the result set of each $R_{f',j}$. From line 16 to line 21, if $R_{f',p}$ is different from $R_{f'+1,p}$, it means the image file f' needs deduplication.

The algorithm complexity is equal to $O(|F|P)$ which looks like a polynomial time with the assumption that calculate processing resource by the number of CPU time slices.

6. EVALUATION

We now consider Online-deduplication performance. As discussed in previous sections, the design of our mechanism includes a number of factors that we expect to impose considerable overheads on performance. Virtual disk address virtualization is provided by the Online-deduplication daemon, which runs in user space in an isolated VM and therefore incurs context-switching on every batch of block requests.

Additionally, our address mapping metadata involves 6-levels B-trees, which risks a dramatic increase in the latency of disk accesses due to seeks on uncached metadata blocks.

There are two questions that this performance analysis attempts to answer. First, what are the overheads that Online-deduplication imposes on the processing of I/O requests? Second, what are the efficiency implications of the virtual machine redundancy free storage that online-deduplication provides? We address these questions in turn, using sequential read and write and PostMark to answer the first and using a combination of micro and macro benchmarks to address the second.

In all tests, we use Dell machines, each node with a 1.86 GHz Xeon E5620 processor, 4 GByte of RAM, and an Intel e1000 GbE network interface. Storage is provided by a western digital exporting an sata ii over 3-gigabits links.We have been developing against XCP 1.5 as a base. One notable modification that we have made to XCP has been to double 4th maximum number of block requests, from 32 to 64, that a guest may issue at any given time, by allocating an additional shared ring page in the split block (blkback) driver. The standard 32-slot rings were shown to be a bottleneck when connecting to Ethernet over a high capacity network.

First, we evaluate the energy efficiency of our demonstration. Considering the difficulty to reappear user behavior in cloud platform, we choose a typical usage scene that booting VM concurrently in a short period to evaluate the energy consumption of VM storage. As comparison, we also record the result of same test with direct connection with our distributed storage system.

Our analysis compares access to the block device in XCP host VM (dom0 in the graphs) and to guest VM with online-deduplication.

Accessing block devices from dom0 has the least overhead, in that there is no extra processing required on block request and dom0 has direct access to the network interface. This configuration is effectively the same as unvirtualized Linux with respect to block performance. In addition, in dom0 tests, the full system RAM and both hyperthreads are available to dom0. In the following cases, the memory and hyper threads are equally divided between dom0 (which acts as the Storage VM5) and a guest VM.

In the direct mode, we access the block device from a guest VM over XCP blkback driver. In this case, the guest runs a block driver that forwards requests over a shared memory ring to a driver (blkback) in dom0, where they are issued to the network stack. Dom0 receives direct access to the relevant guest pages, so there is no copy overhead, but this case does incur a world switch between the client VM and dom0 for each batch of requests.

Finally, in the case of online-deduplication, the configuration is similar to the direct case, but when requests arrive at the dom0 kernel module (blktap instead of blkback), they are passed on to the online-deduplication daemon running in user space. Our system issues reads and writes to the Linux kernel using Linuxs asynchronous I/O interface (libaio), which are then issued to the sata bus or networks.

6.1. Energy Consumption

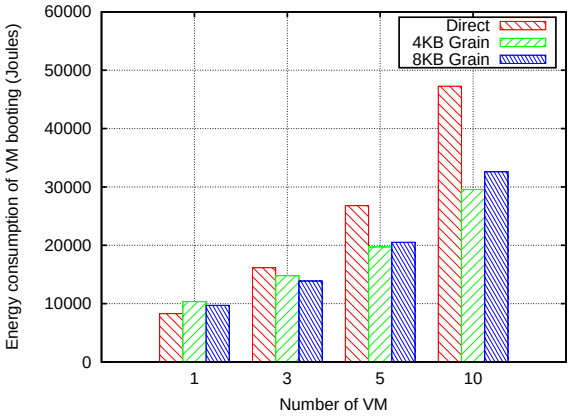

FIGURE 4. The energy consumption by VM booting.

VM booting is a very frequent operation in existed cloud service providers. From anecdotal analysis of EC2, based on decoding the instance ID, concluded that $O(10^4)$ new VM instances are requested each day [12]. Considering the convenience and practicality, we measure the energy consumption of VM booting instead of other benchmark applications which are very rare in real use of cloud services. Considering the performance limitation of our testbed hardware, we set the number of concurrent VM booting from 1 to 10 and record the time of beginning and finish booting to get the interval of all VM finishing booting. As comparison, we use the same VM instances with direct connection storage and record same interval. We execute the test 5 times and record the average interval as the result.

From the test result shown in FIGURE 4, with the computing overload of the deduplication, we find the energy consumption of our system is almost the same with the general storage system. With increase of VM number, comparing the linear increasing consumption without deduplication, enhanced energy efficiency of our system is obviously observed. When using 4KB data block size, the energy consumption of booting 10 VM in our system is only 62 % of the comparing system. With 8KB data block size, it costs a bit more energy with 66 % of comparing system. When the number of VM is less than 5, since small block size needs more computing in deduplication, using 8KB block size in our storage performs better energy efficiency than 4KB block size. When the number of VM increased, considering higher deduplication ratio of the smaller block size, less data transferring by using a small block

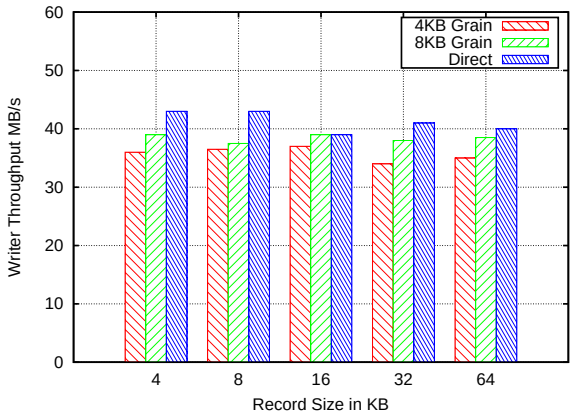

FIGURE 5. System storage performance against a local disk as reported by IOZone Writer.

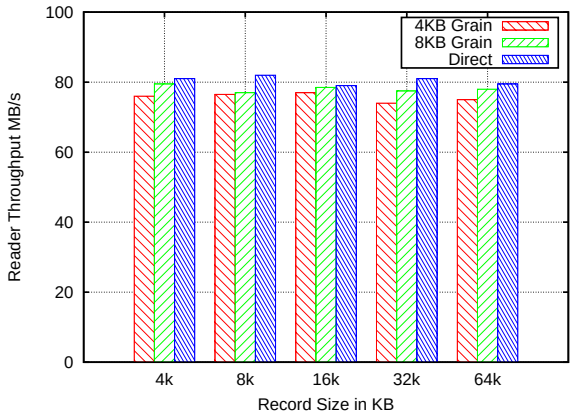

FIGURE 6. System storage performance against a local disk as reported by IOZone Reader.

size get a better energy efficiency.

6.2. Sequential I/O

For each of the three possible configurations, we ran IOZone [13] five in succession. The first run provided 4KB grain data points, while the second allows online-deduplication to extent its block size to 8KB. As shown in FIGURE 5, the better write performance in the bigger grain case demonstrates that online-deduplication is unable to maintain write performance near the effective line speed of the direct access. Our system performance is within 65% of dom0. At the same time, the 60% performance degradation in the 8KB grain case underscores the importance of block size in online-deduplication, as doing so limits the overheads involved in B-tree traversal. As we have focused our efforts to date on tuning the write path, we have not yet sought aggressive optimizations for read operations. This is apparent in the IOZone test, as we can see read performance slipping to more than 14% lower than that of our non-virtualized dom0 configuration.

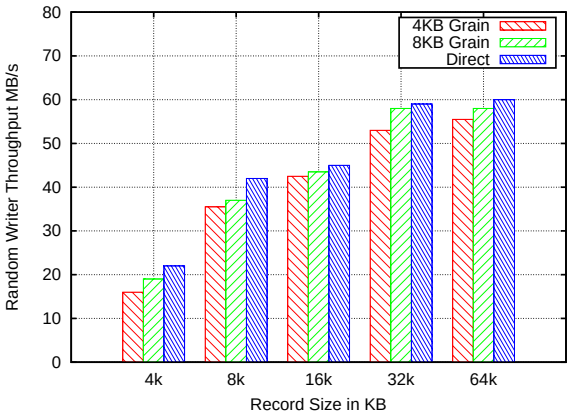

FIGURE 7. System storage performance against a local disk as reported by IOZone random Writer.

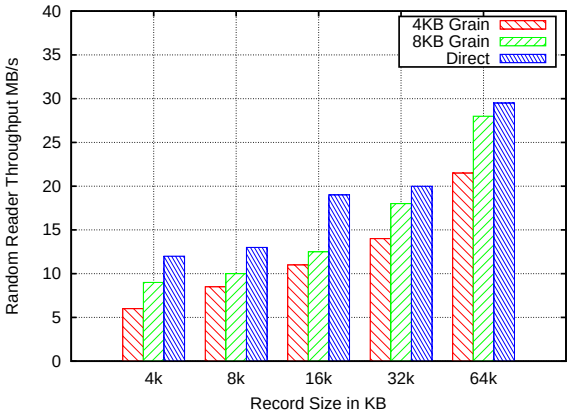

FIGURE 8. System storage performance against a local disk as reported by IOZone random Reader.

6.3. Random I/O

FIGURE 7 and FIGURE 8 show the results of running random IOZone test with the online-deduplication and directly attached configuration. Unlike the sequential I/O, the cache in random I/O test is much colder while the impact of data reform in deduplication is less than warm cache in sequential I/O test. However, when the data size random become large, the random I/O performance of Domain 0 is getting closer to the sequential I/O while the online-deduplication remains almost the same with its discontinuous data placement.

7. RELATED WORKS

In this section, we firstly introduce some related works about energy efficiency storage, VM storage and data deduplication.

7.1. General Storage Consumption

Sleeping idle devices are the most deeply researched method in general storage system especially the hard disk (HDD) devices. In general data center

storage environment, Zedlewski et al. [14] model the HDD consumption to accurately estimate the energy consumed by sub-components of a disk request. Based on their work which points out the importance of analysis of the power mode transition, many works worked on HDD spin-up and spin-down. Meanwhile, Li. et al. [15] analyze the pros and cons of spinning down disks and address the sleeping time of the disk to reduce energy consumption. To optimize the problem of HDD spinning down, Chung et al. [16] propose an adaptive approach which slides windows with two-dimensional linear interpolation of power managing a single disk.

Cache deployed in storage I/O procedure is another efficient method to reduce energy consumption. Zhu et al. [17] propose a set of cache management methods to reduce the disk energy consumption, which focuses on cache layer, and studies of many cache methodologies. Meanwhile, many previous works study deeply about buffering writes and performing periodic updates in the storage system.

Using a timer to monitor the disk access traffic is a more simple method for scheduling HDD sleeping. By adapting to the user's access, Douglis et al. [18] vary the spin down threshold dynamically to reduce the disk spin ups. Golding et al. [19] design several idle detection algorithms to decrease disk energy consumption. With more aggressive dynamic predictors, the time to sleep HDD is much longer than timer in scheduling. Accordingly [16, 20, 21], using the HDD access information of recent history to predict the idle period for sleeping idle HDD.

7.2. VM storage with energy efficiency

In VM storage, even the structure is different, some researchers use a similar method to reduce energy consumption with original storage system. Ye et al. [22] propose a mechanism to improve HDD energy dissipation by adjustment of buffer cache flush to save disk energy and implement their design in the virtual machine monitor (VMM). Stoess et al. [23] also insert a framework in VMM to distinguish two different power states, which is capable of enforcing power limits of guest VM. Nathuji et al. [24] propose similar management in VMM to support online power management, which reduces 34% power consumption in VM storage.

Using VM migration and placement is another methodology based on the facilities of virtualization to improve energy efficiency in cloud data center. Verma et al. [3] propose a dynamic consolidation of storage based on VM migration to enhance the energy efficiency. Dong et al. [25] design a VM placement algorithm in cloud data center to save energy. In their VM placement method, storage energy is considered as a main factor of resources rather than a special discussing. Shailesh S.Deore et al. [26] introduced an energy efficient scheduling technology to avoiding changing the status of idle node to power on to reducing the energy consumption including the storage system. Anton Beloglazov et al. [27] propose heuristic algorithms to optimize the energy efficiency after modeling the resource consumption (CPU, disk storage and network interface) in cloud computing. Xin Li et al. [28] proposed a virtual machine placement algorithm EAGLE, which can balance the utilization of multidimensional resources, reduce the number of running PMs, and thus lower the energy consumption while it is not an efficient method for reducing storage energy consumption since the workloads are not changed. Ahmed Sallem et al.[29] proposed a migration policy for multi-object optimization strategy, in which energy efficiency is one of objects in this work. Since it is hard to reduce workloads by migration, this work can not improve energy efficiency in the ordinary centralized storage system.

7.3. Deduplication System

Venti [30], EMC Avamar [31], Pergamum [32] and DataDomain [33] are typical deduplication storage systems. Venti, Pergamum and Avamar target data archiving, whereas DataDomain is designed to store backup data. Venti prototype and Centera support duplicate elimination, respectively, on fixed block size and entire file level. Venti, archiving data as a network storage system, focuses on saving storage space. It stores duplicated copies of fixed size data blocks only once. Venti reports a reduction of around 30% in the size of the data sets employing this method.

EMC Avamar, as an enterprise storage server, it takes a way to implement deduplication by software agents. These agents work background to deduplicate data through network with fingerprints. Each agent can recognize message processed by sending fingerprint to the central deduplication system and comparing the records existed. Without the agents, deduplication can only be used on independent machines.

Pergamum is a distributed intelligent disk based storage system. As a disk based archive solution, it must be easily scalable in capacity, performance and time. With NVRAM [34] in each node to store data signatures, metadata and other small items, Pergamum allows to defer writing, metadata requests and inter-disk data verification to be performed while the disk is powered off. Although Pergamum has done some work on deduplication, the duplicate elimination is not supported well in the system, which means in Pergamum the storage capacity cannot be recycled even it knows the data have been stored before.

8. CONCLUSIONS

In this paper, we propose an energy efficient VM storage system in cloud data center that uses online-deduplication mechanism to reduce the energy

consumption from redundant data. We design online-deduplication mechanism that eliminates the redundant data before stored in storage system. To support this mechanism, we design a two layer storage structure both in virtualization servers and storage units. After that, we implement a demonstration system based our design. We also study a deduplication selection problem to minimum the energy consumption of the VM storage system with limited deduplication processing resources. Finally, the demonstration system evaluations are conducted to show that the proposed system can significantly reduce the energy consumption with negligible overload.

REFERENCES

[1] Dong, M., Li, H., Ota, K., and Zhu, H. (2014). Hvsto: Efficient privacy preserving hybrid storage in cloud data center. `http://arxiv.org/abs/1405.6200`.

[2] Ibrahim, H., Ilinca, A., and Perron, J. (2008) Energy storage systems haracteristics and comparisons. *Renewable and Sustainable Energy Reviews*, **12**, 1221–1250.

[3] Akshat, V., Ricardo, K., Luis, U., and Raju, R. (2010) Srcmap: Energy proportional storage using dynamic consolidation. *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, Berkeley, CA, USA, pp. 20–20. USENIX Association.

[4] Berl, A., Gelenbe, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M. Q., and Pentikousis, K. (2010) Energy-efficient cloud computing. *The Computer Journal*, **53**, 1045–1051.

[5] Zhu, Q., David, F. M., Devaraj, C. F., Li, Z., Zhou, Y., and Cao, P. (2004) Reducing energy consumption of disk storage using power-aware cache management. *Proceedings of the 10th International Symposium on High Performance Computer Architecture*, Washington, DC, USA HPCA '04, pp. 118–118. IEEE Computer Society.

[6] Bianchini, R. and Rajamony, R. (2004) Power and energy management for server systems. *Computer*, **37**, 68–74.

[7] Citrix. Xen cloud platform - advanced virtualization infrastructure for the clouds. `http://www.xen.org/products/cloudxen.htm`.

[8] Liao, X., Jin, H., Hu, L., and Liu, H. (2010) Towards virtualized desktop environment. *Concurrency and Computation: Practice & Experience*, **22**, 419–440.

[9] Wang, X., Feng, D., Lai, X., and Yu, H. (2004). Collisions for hash functions md4, md5, haval-128 and ripemd. Cryptology ePrint Archive. `http://eprint.iacr.org/2004/199`.

[10] Chang, Y.-C., Chang, Y.-W., Wu, G.-M., and Wu, S.-W. (2000) B*-trees: a new representation for non-slicing floorplans. *Proceedings of the 37th Design Automation Conference.*, pp. 458–463. ACM.

[11] Andonov, R., Poirriez, V., and Rajopadhye, S. (2000) Unbounded knapsack problem: Dynamic programming revisited. *European Journal of Operational Research*, **123**, 394–407.

[12] Rosen, G. (2009). Anatomy of an amazon ec2 resource id. `http://www.jackofallclouds.com/2009/09/anatomy-of-an-amazon-ec2-resource-id/`.

[13] Norcott, W. D. and Capps, D. (2006). Iozone filesystem benchmark. `www.iozone.org`.

[14] Zedlewski, J., Sobti, S., Garg, N., Zheng, F., Krishnamurthy, A., and Wang, R. (2003) Modeling hard-disk power consumption. *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, Berkeley, CA, USA, pp. 217–230. USENIX Association.

[15] Li, K., Kumpf, R., Horton, P., and Anderson, T. (1994) A quantitative analysis of disk drive power management in portable computers. *Proceedings of the USENIX 1994 Technical Conference*, Berkeley, CA, USA, pp. 22–22. USENIX Association.

[16] Chung, E.-Y., Benini, L., Bogiolo, A., and De Micheli, G. (1999) Dynamic power management for non-stationary service requests. *Proceedings of the Conference on Design, Automation and Test in Europe*, New York, NY, USA. ACM.

[17] Qingbo, Z., M., D. F., F., D. C., Zhenmin, L., Yuanyuan, Z., and Pei, C. (2004) Reducing energy consumption of disk storage using power-aware cache management. *Proceedings of the 10th International Symposium on High Performance Computer Architecture*, Washington, DC, USA, pp. 118–118. IEEE Computer Society.

[18] Douglis, F., Krishnan, P., and Bershad, B. N. (1995) Adaptive disk spin-down policies for mobile computers. *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*, Berkeley, CA, USA, pp. 121–137. USENIX Association.

[19] Golding, R., Bosch, P., Staelin, C., Sullivan, T., and Wilkes, J. (1995) Idleness is not sloth. *Proceedings of the USENIX 1995 Technical Conference*, Berkeley, CA, USA, pp. 17–17. USENIX Association.

[20] Hwang, C.-H. and Wu, A. C.-H. (2000) A predictive system shutdown method for energy saving of event-driven computation. *ACM Transactions on Design Automation of Electronic Systems*, **5**, 226–241.

[21] Srivastava, M. B., Chandrakasan, A. P., and Brodersen, R. W. (1996) Predictive system shutdown and other architectural techniques for energy efficient programmable computation. *IEEE Transactions on Very Large Scale Integration Systems*, **4**, 42–55.

[22] Ye, L., Lu, G., Kumar, S., Gniady, C., and Hartman, J. H. (2010) Energy-efficient storage in virtual machine environments. *Proceedings of the 6th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, New York, NY, USA, pp. 75–84. ACM.

[23] Stoess, J., Lang, C., and Bellosa, F. (2007) Energy management for hypervisor-based virtual machines. *Proceedings of 2007 USENIX Annual Technical Conference*, Berkeley, CA, USA, pp. 1:1–1:14. USENIX Association.

[24] Nathuji, R. and Schwan, K. (2007) Virtualpower: Coordinated power management in virtualized enterprise systems. *Proceedings of the 21th ACM SIGOPS Symposium on Operating Systems Principles*, New York, NY, USA, pp. 265–278. ACM.

[25] Dong, J., Jin, X., Wang, H., Li, Y., Zhang, P., and Cheng, S. (2013) Energy-saving virtual machine placement in cloud data centers. *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 618–624. IEEE Computer Society.

[26] S.Deore, S. and Patil, A. N. (2012) Energy-efficient scheduling scheme for virtual machines in cloud computing. *International Journal of Computer Applications*, **56**, 19–25. Published by Foundation of Computer Science, New York, USA.

[27] Beloglazov, A., Abawajy, J., and Buyya, R. (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, **28**, 755 – 768. Special Section: Energy efficiency in large-scale distributed systems.

[28] Li, X., Qian, Z., Lu, S., and Wu, J. (2013) Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. *Mathematical and Computer Modelling*, **58**, 1222–1235.

[29] Sallam, A. and Li, K. (2014) A multi-objective virtual machine migration policy in cloud systems. *The Computer Journal*, **57**, 195–204.

[30] Quinlan, S. and Dorward, S. (2002) Venti: A new approach to archival data storage. *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, Berkeley, CA, USA. USENIX Association.

[31] Geer, D. (2008) Reducing the storage burden via data deduplication. *Computer*, **41**, 15–17.

[32] Storer, M. W., Greenan, K., Miller, E. L., and Voruganti, K. (2008) Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage. *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, February, pp. 1–16. USENIX Association.

[33] Tax, D. M. J. and Duin, R. P. W. (1999) Data domain description using support vectors. *Proceedings of the 7th European Symposium on Artificial Neural Networks*, pp. 251–256.

[34] Onishi, S. (1998). Non-volatile random access memory. US Patent 5,708,284.