2016

A Fresh Perspective on Total Cost of Ownership Models for Flash Storage in Datacenters

Zhengyu Yang, *Northeastern University*
Manu Awasthi
Mrinmoy Ghosh
Ningfang Mi, *Northeastern University*

A Fresh Perspective on Total Cost of Ownership Models for Flash Storage in Datacenters

Zhengyu Yang*, Manu Awasthi†, Mrinmoy Ghosh†, and Ningfang Mi*
* Dept. of Electrical & Computer Engineering, Northeastern University, 360 Huntington Ave., Boston, MA 02115
† Samsung Semiconductor Inc., San Jose, CA 95134

***Abstract*—Recently, adoption of Flash based devices has become increasingly common in all forms of computing devices. Flash devices have started to become more economically viable for large storage installations like datacenters, where metrics like Total Cost of Ownership (TCO) are of paramount importance. Flash devices suffer from write amplification (WA), which, if unaccounted, can substantially increase the TCO of a storage system. In this paper, we develop a TCO model for Flash storage devices, and then plug a Write Amplification (WA) model of NVMe SSDs we build based on empirical data into this TCO model. Our new WA model accounts for workload characteristics like write rate and percentage of sequential writes. Furthermore, using both the TCO and WA models as the optimization criterion, we design new Flash resource management schemes (MINTCO) to guide datacenter managers to make workload allocation decisions under the consideration of TCO for SSDs. Experimental results show that MINTCO can reduce the TCO and keep relatively high throughput and space utilization of the entire datacenter storage.**

***Keywords*—Flash Resource Management, Total Cost of Ownership Model, SSD Write Amplification, NVMe, Wearout Prediction, Workload Sequentiality Pattern**

I. INTRODUCTION

The world has entered the era of "Big Data", when large amount of data is being collected from a variety of sources, including computing devices of all types, shapes and forms. This data is then being pushed back to large, back-end datacenters where it is processed to extract relevant information. As a result of this transformation, a large number of server-side applications are becoming increasingly I/O intensive. Furthermore, with the amount of data being gathered increasing with every passing day, the pressure on the I/O subsystem will continue to keep on increasing [1].

To handle this high I/O traffic, datacenter servers are being equipped with the best possible hardware available encompassing compute, memory, networking and storage domains. Traditionally, I/O has been handled by hard disk drives (HDDs). HDDs have the benefit of providing an excellent economic value ($/GB), but being built with mechanical, moving parts, they suffer from inherent physical throughput limitations, especially for random I/Os. To counter these performance limitations, solid state devices (SSDs) have recently begun to emerge as a viable storage alternative to HDDs. In the recent past, SSDs have gained widespread adoption owing to reduced costs from the economies of scale. Datacenters, especially popular public cloud providers (e.g., [2], [3]) have been at the forefront of adopting Flash technology.

Nevertheless, during this revolutionary change in cloud storage systems, Flash based SSDs face two major concerns: cost and write amplification (WA). Firstly, the costs of owning (purchasing and maintaining) SSDs can still be very high. Balancing the trade-off between performance and economy is still an uphill battle. Currently, Total Cost of Ownership (TCO), comprising of two major costs: (i.e., Capital and Operating Expenditures), remains a popular metric. However, only a few prior studies have focused on the TCO of SSDs in datacenters, especially with the consideration of the cost of SSD's wearout.

Secondly, SSDs have limited write cycles and also suffer from Write Amplification (WA) which is caused by a number of factors specific to Flash devices including erase-before-rewrite, background garbage collection, and wear leveling. In fact, the WA of an SSD is a direct function of the I/O traffic it experiences. The I/O traffic, in turn, comprises of a number of different factors like the fraction of writes (as opposed to reads), the average size of I/O requests, the arrival rate of I/Os, and the ratio of sequential I/O patterns (as opposed to random I/O) in the overall I/O stream. Greater WA can significantly reduce the lifetime and increase the ownership cost of Flash devices.

Therefore, in this work, we focus on addressing the above two concerns by investigating the relationship between workload patterns and WA and then leveraging the relationship to develop new TCO models. In our experiments, we found that workloads with different sequential ratios have varying write amplifications even on the same SSD, which changes the lifetime of the device and eventually affects the TCO. We are thus motivated to evaluate storage systems from a cost perspective that includes many dimensions such as maintenance and purchase cost, device wearout, workload characteristics, and total data amount that can be written to the disk, etc. Therefore, in this paper, we make the following contributions to achieve this goal.

- We conduct real experiments to measure and characterize the write amplification under different workloads, and reveal the relationship between write amplification and workload sequential ratio for each disk with fixed Flash Translation Layer(FTL) specs.
- We propose a new TCO model while considering multiple factors like SSD lifetime, workload sequentiality, write wearout and the total writes.
- We propose statistical approaches for calculating components that are essential for computing the TCO but cannot

This work was completed during Zhengyu Yang's internship at Samsung Semiconductor Inc., and was partially supported by National Science Foundation Career Award CNS-1452751 and AFOSR Grant FA9550-14-1-0160.

9781-5090-1445-3/16$31.00

(practically) be measured from SSDs during runtime, such as write amplification and wearout of each SSD.
- Based on our TCO model, we develop a set of new online adaptive Flash allocation managers called "MINTCO", which leverage our TCO model to dynamically assign workloads to the SSD disk pool. The goals of MINTCO are: (1) to minimize the TCO, (2) to maximize client throughput as many as possible, and (3) to balance the load among SSD devices and best utilize SSD resources.

Lastly, we evaluate our new models and approaches using real world trace-driven simulation. Our experimental results show that MINTCO can reduce the TCO by up to 90.47% compared to other traditional algorithms. Meanwhile, it guarantees relatively high throughput and spatial utilization of the entire SSD-based datacenter.

The remainder of this paper is organized as follows. Sec. II investigates the cause of write amplification on SSDs. Sec. III presents the details of our TCO models. Sec. IV proposes two versions of our MINTCO allocation algorithms. Sec. V measures the WAF of NVMe disks under different workload patterns, and evaluates our allocation algorithms. Sec. VI describes the related work. Finally, we summarize the paper and discuss the limitations and future work plan of this research in Sec. VII.

II. Write Amplification Factor

Write amplification factor ("WAF", henceforth referred to as "A" and "WA") is a commonly used metric to measure the write amplification degree. WAF is an undesirable phenomenon associated with Flash devices where the actual amount of data written to the device is larger than the logical amount of data written by a workload. We define WAF as the ratio between the total physical write data written by the SSD and the total logical data written by the workload: $A = \frac{W_P}{W_L}$, where W_L denotes the logical write amount (in bytes) and W_P denotes the physical, device-level I/O writes as seen by the SSD. Fig. 1 illustrates the logical and physical writes all the way from the application, through the OS, to the SSD. Large values of A lead to increase I/O latency, shorten the SSD's working lifetime, and increase power consumption.

Flash devices have an unique property that they cannot be re-written unless they have been erased. Also, the minimum granularity of an erase operation is in the order of MBs (e.g., blocks), while the granularity of writes is much smaller, in the order of KBs (e.g., pages). Meanwhile, Flash devices have limited write life cycles. Thus, for the purpose of wear-leveling, the logical address space in Flash devices is dynamically mapped to the physical space and the mapping changes with every write. Flash devices have a software called FTL (Flash Translation Layer) running on them to manage the erase before re-write and wear-leveling requirements. The FTLs have to schedule periodic garbage collection events to de-fragment their write data. These garbage collection events can lead to extra writes that have not been generated by the host. Additionally, SSD reserves a user-invisible space (i.e., over-provision), which is helpful to reduce the WAF during these above-mentioned events to some extent. However, since Flash devices have limited write-erase cycles, the mismatch between the two (logical and physical) types of writes can still cause the SSD to fail much more quickly than expected. On the other hand, besides these device hardware-related factors, write amplification is also affected by I/O workload-related factors,

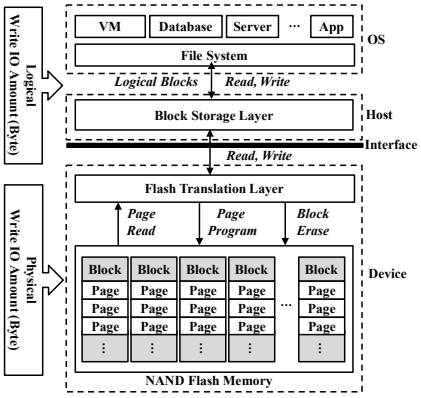

Fig. 1: An example of I/O Path from OS to device.

such as mounted file systems and workload traffic patterns. In this paper, the scenario we are investigating is that all hardware specs of SSDs are fixed in the datacenter after deployment, and we are not aiming to change their FTL algorithms. Therefore, we mainly focus on the impact of different workload patterns to the WAF under SSDs.

The existing analytical models [4], [5] for WAF build the relationship between workload characteristics and WAF based on the different garbage collection policies (i.e., cleaning algorithms) and the impacts of the hot and cold data distribution. However, these models ignore a factor that is becoming increasingly important, especially in the NoSQL database community, traffic patterns in terms of sequential and random ratio experienced by the SSD [6]. With the proliferation of log structured merge tree (LSM tree) based NoSQL databases, there is a lot of uptick in the amount of sequential traffic being sent to the SSDs. LSM-tree based databases capture all the writes into a large in-memory buffer. When the buffer is full, it is flushed to disk as a large multi-gigabyte sequential write. Another similar case is write-intensive workloads that execute within virtual machines – most of the write traffic to the SSD is usually sent out as large, sequential writes [7]. Hence, it is becoming increasingly essential to understand the WAF, performance of SSDs, device worn-out, and most importantly, the total owning cost of datacenters from a workload-centric view.

III. TCO Models of SSD-Intensive Datacenters

TCO of a datacenter is a mix of a large number of items. Broadly speaking, these items can be broken down into two major categories: (1) Capital Expenditure (CapEx), and (2) Operating Expenditure (OpEx). Capital Expenditure refers to the amount of money that needs to be spent in setting up a facility. These include the cost of buying individual components of the servers, power supplies, racks that house the servers, among other things. OpEx, on the other hand, is the amount of money that is spent in the day-to-day operation of a datacenter. Examples of OpEx include electricity costs and personnel costs (required for maintaining the datacenter). CapEx is traditionally a large, one time expenditure [8] while OpEx consists of small(er), recurring expenditures. In this section, we develop a TCO model for an SSD intensive datacenter of the future, based on the characteristics of the workloads that are scheduled on to those SSD devices. Our TCO model focuses specifically on the costs related to acquiring and maintaining SSDs in a datacenter.

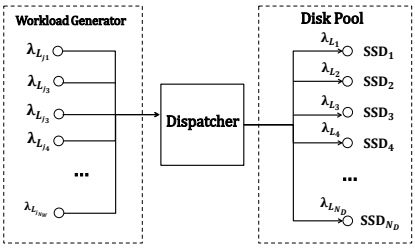

Fig. 2: Model of a datacenter storage system.

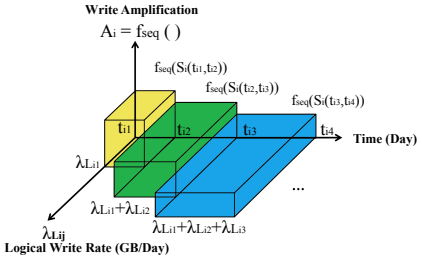

Fig. 3: An example of write wearout count estimation.

A. Datacenter Storage Model

First, we briefly explain our assumptions about datacenters, their workloads and storage systems. We assume the datacenter to be a large pool of SSD devices. This helps us abstract the problem of modeling SSDs from a per-server resource to a pool of datacenter-wide resources. We then model the storage system of such a datacenter as a *workload-to-disk allocation problem*, as shown in Fig. 2. In this model, we have a pool of N_D SSDs as shown in Fig. 2. Meanwhile, there are N_W applications (workloads) that submit I/O requests with logical write rates λ_{LJ_i} (where $1 \leq i \leq N_W$, and "LJ" stands for "logical" and "job"), as seen in the left hand box of Fig. 2. To complete the connection between I/O requests and the SSDs, we assume an allocation algorithm that is used by the dispatcher to assign I/O workloads to different SSDs. Each workload has its own characteristic, and arrives at the dispatcher at different times. Multiple workloads can be assigned to the same disk as long as the capacity (e.g., space and throughput) of the disk is sufficient, such that the logical write rate (λ_{L_i}) to disk i is the summation of logical write rates from the workloads in the set $\mathbb{J}_i$ that are allocated to that SSD, i.e., $\lambda_{L_i} = \sum_{j \in \mathbb{J}_i} \lambda_{LJ_j}$. We summarize our main assumptions as follows.

1) ***I/O streams with certain properties****:* "Workload" is defined as an endless logical I/O stream issued by applications. Particularly, from a long-term view (e.g., years), characteristics of workloads, such as sequential ratio, daily write rate, read-write ratio, working set size, re-access ratio, can be abstracted as (almost) fixed values. Workloads may arrive to the datacenter at different times. Once a workload arrives, the dispatcher assigns it to a certain disk, and the disk will execute this workload until the disk is "dead" (i.e., SSD write cycle limit is reached), or the workload finishes. We ignore the overhead (such as time and energy consumption) during the workload deployment.

2) ***Isolation among multiple workloads on a single disk****:* Multiple workloads can be assigned to a single SSD, and have separate and independent working sets (i.e., address spaces and segments are isolated). Therefore, the cross-workloads effects along I/O path due to interleaving working sets are negligible.

3) ***SSD, as an blackbox, can be reflected by WA****:* We use the WA model to capture the behavior of an SSD under a workload with a specific I/O pattern. Thus, we estimate the WAF of each disk by using the sequentiality information of multiple workloads that are concurrently executing at a particular SSD. The write wearout of each SSD can further be estimated using the runtime status of that SSD's WA.

B. TCO Model

Owning and maintaining a low cost SSD-intensive datacenter is critically important. TCO has been widely adopted to evaluate and assess storage subsystem solutions. However, to the best of our knowledge, there is no standard formula for calculating the TCO of the SSD-intensive storage subsystem. In order to comprehensively access the expenditure of a datacenter, a generic TCO model should consider purchasing and maintenance costs, service time, served I/O amount and device wearout. We present the following models to calculate the TCO. As we mentioned, two major types of costs: CapEx (C_{I_i}) and OpEx (C_{M_i}) are considered in the basic TCO model. In detail, $C_{I_i} = C_{Purchase_i} + C_{Setup_i}$ and $C'_{M_i} = C'_{Power_i} + C'_{Labor_i}$, where $C_{Purchase_i}$ and C_{Setup_i} are one-time cost ($) of device purchase and device setup, and C'_{Power_i} and C'_{Labor_i} are power and maintenance labor cost rate ($/day). Although CapEx ($C'_{I_i}$) is one time cost, OpEx (C'_M) is a daily rate and the TCO should be depends on the amount of time that an SSD has been used. Therefore, we need to attach a notion of *time* to TCO. Assume we know the expected life time (T_{Lf_i}) of each disk (i.e., $T_{Lf_i} = T_{D_i} - T_{I_i}$, where T_{D_i} and T_{I_i} are the time when the disk i is completely worn out and the time when it was started accepting its first request, respectively), the total cost for purchasing and maintaining a pool of SSDs can be calculated as:

$$TCO = \sum_{i=1}^{N_D} (C_{I_i} + C'_{M_i} \cdot T_{Lf_i}), \tag{1}$$

where N_D is the number of disks in the pool. Fig. 4(a) also illustrates an example from time stamp view, where I/O workloads keep arriving and thus the physical write rate of disk i increases accordingly. However, Eq. 1 does not reflect SSD wearout at all, which is highly coupled with the workload arrival distribution. For instance, in a datacenter consisted of the same type of SSDs, the SSD running workloads with the lowest physical write rate may always have the highest TCO value (i.e., its $C'_{M_i} \cdot T_{Lf_i}$ is the greatest among all) according to Eq. 1, since this SSD is probably the last one to be worn out. However, this SSD may have a larger WAF due to the workload pattern, while others that died earlier may have smaller WAFs and can serve more logical write I/O amounts under the same cost. Therefore, to conduct a fair and meaningful comparison, we introduce the data-averaged TCO rate (TCO') from the perspective of the cost vs. the total amount of (logical) data served (written) to an SSD as follows.

$$TCO' = \frac{\sum_{i=1}^{N_D} (C_{I_i} + C'_{M_i} \cdot T_{Lf_i})}{\sum_{j=1}^{N_W} D_j}, \tag{2}$$

where $\sum_{j=1}^{N_W} D_j$ is the total *logical* data write amount for all N_W workloads. Again, we use logical writes as a proxy for physical writes not only because the former is much easier to obtain for most workloads, but also because by being normalized by the logical writes, the TCO' is able to reflect

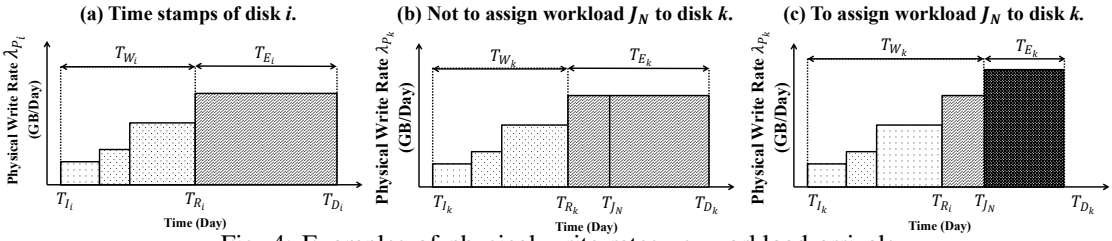

Fig. 4: Examples of physical write rates vs. workload arrivals.

the WAF and judge the disk-level wear leveling performance of different allocation algorithms.

C. Calibrating TCO Models

The models developed in the prior section have all assumed that certain parameters for TCO calculation (e.g., total logical data write amount, expected lifetime, etc.) are readily available or measurable. However, it is impractical to measure some parameters that are necessary in our TCO models. In this section, we propose a mathematical approach of estimating those unmeasurable parameters.

*1) **Total Logical Data Writes** $\sum_{j=1}^{N_W} D_j$:* Given workload j's logical write rate λ_{L_j}, arrival time T_{A_j} and estimated time of death ($T_{D(j)}$) of workload j 's host disk $D(j)$, we can calculate the total amount of data written by all the jobs over their course of execution as: $\sum_{j=1}^{N_W} D_j = \sum_{j=1}^{N_W} \lambda_{L_j}(T_{D(j)} - T_{A_j})$, where λ_{L_j} is the logical data write rate of workload j. The only unknown parameter left is $T_{D(j)}$, which can be obtained by calculating each host disk's expected lifetime.

*2) **Expected Lifetime** T_{Lf_i}:* The lifetime of a disk depends not only on the write traffic from the currently executing jobs, but also on those jobs that have already been deployed on the disk. Furthermore, we also need to account for the effects of the *combined* write traffic of the workloads that are concurrently executing on a disk. As shown in Fig. 4(a), the lifetime of disk i is the period from T_{I_i} to T_{D_i}. We further split the lifetime into two phases: (1) accumulated work epoch (T_{W_i}) until the last workload arrives, i.e., $T_{W_i} = T_{R_i} - T_{I_i}$, where T_{R_i} is the assigned time of the most recent workload; and (2) expected work lifetime (T_{E_i}) from T_{R_i} to the expected death time, i.e., $T_{E_i} = T_{D_i} - T_{R_i}$. The former is easy to monitor, and the latter is further estimated as the available remaining write cycles of disk i divided by the physical data write rate (λ_{P_i}) of disk i from T_{Ri}. Moreover, λ_{P_i} can be calculated as disk i's logical data write rate (λ_{L_i}) times disk i's WAF (A_i). Thus, we have $T_{Lf_i} = T_{D_i} - T_{I_i} = T_{W_i} + T_{E_i} = (T_{R_i} - T_{I_i}) + \frac{W_i - w_i}{\lambda_{P_i}} = (T_{R_i} - T_{I_i}) + \frac{W_i - w_i}{\lambda_{L_i} \cdot A_i} = (T_{R_i} - T_{I_i}) + \frac{W_i - w_i}{\lambda_{L_i} \cdot f_{seq}(S_i)}$, where A_i, W_i, w_i and S_i are the WAF function, the total write limit, current write count (wearout), and sequential ratio of all running workloads of disk i, respectively. Since the SSDs' hardware are fixed, we denote A_i as a function of workload's write I/O sequential ratio (f_{seq}) of disk i, which will be validated and regressed in our experimental section (Sec. V-A5). In fact, we can plug any WAF model into this TCO model. As of now, we also know T_{R_i}, T_{I_i} and W_i, and what we need to estimate next are the remaining parameters, i.e., λ_{L_i}, S_i and w_i.

*3) **Logical Write Rate of Workloads on Disk** λ_{L_i}:* For disk i, its logical write rate λ_{L_i} should be the sum of all its assigned workloads' logical write rates, i.e., $\lambda_{L_i} = \sum_{j \in \mathbb{J}_i} \lambda_{L_{ij}}$, where $\mathbb{J}_i$ is the set of workloads running on disk i. Notice that there is an extreme case during the very early stage where no workloads have been assigned to the disk i (i.e., $\mathbb{J}_i = \emptyset$), such that $\frac{W_i - w_i}{\lambda_{L_i} \cdot f_{seq}(S_i)}$ becomes infinite. To avoid such an extreme case, we conduct a warming up process that assigns at least one workload to each disk. Only after this warming up phase is done, we start to calculate T_{Lf_i}.

*4) **Sequential Ratio of Workloads on Disk** S_i:* In order to calculate the write amplification A_i in Sec. III-C2, we need to know the sequential ratio of multiple workloads that are assigned to one disk. Unlike the logical write rate, the combined sequential ratio of multiple workloads is not equal to the sum of sequential ratios of all workloads. Our estimating solution is to assign a weight to each workload stream's sequential ratio and set the weight equal to the workload's logical data write rate. Hence, for multiple workloads running on the disk, we can calculate the overall sequential ratio as: $S_i = \frac{\sum_{j \in \mathbb{J}_i} \lambda_{L_{ij}} S_{ij}}{\sum_{j \in \mathbb{J}_i} \lambda_{L_{ij}}}$. where $\lambda_{L_{ij}}$ and S_{ij} are the logical write rate and sequential ratio of jth workload running on disk i.

*5) **Write Wearout Count of Disk** w_i:* The last item we need to estimate is the current physical write count w_i (in Sec. III-C2) inside each SSD device. It is hard to exactly measure the overall write count of an SSD during its lifetime. However, we can estimate the current write count by adding the estimated write counts of all the workloads over all past epochs. For each epoch, we multiply the total logical write rate by the corresponding WAF to get the physical write rate. By iterating this process for all epochs, we can finally get the total write wearout count for each disk. Fig. 3 shows a simple example of estimating a disk's write wearout when there are multiple workloads executing on disk i. Each brink represents an epoch, which is bounded by its workloads' allocation times. The volume of all these brinks gives the total write wearout count (w_i) for disk i. To calculate w_i, we further convert above-mentioned λ_{L_i} and S_i to the total logical data write rate function and the sequential ratio function during each epoch $[t_{ix}, t_{i(x+1)})$, respectively: $\lambda_{L_i}(t_{ix}, t_{i(x+1)}) = \sum_{j \in \mathbb{J}_i(t_{ix}, t_{i(x+1)})} \lambda_{L_{ij}}$, and $S_i(t_{ix}, t_{i(x+1)}) = \frac{\sum_{j \in \mathbb{J}_i(t_{ix}, t_{i(x+1)})} \lambda_{L_{ij}} S_{ij}}{\sum_{j \in \mathbb{J}_i(t_{ix}, t_{i(x+1)})} \lambda_{L_{ij}}}$, where x is the number of workloads executing on disk i, and t_{ix} is the arrival time of disk i's x^{th} workload. $\mathbb{J}_i(t_{ix}, t_{i(x+1)})$ is the set of workloads running on disk i during $t_i x$ and $t_{i(x+1)}$ epoch. $\lambda_{L_{ij}}$ and S_{ij} are the write rate and sequential ratio of j^{th} workload in $\mathbb{J}_i(t_{ix}, t_{i(x+1)})$. Therefore, wearout w_i can be calculated as: $w_i = \sum_{t_{ix} \in T_i} [\lambda_{L_i}(t_{ix}, t_{i(x+1)}) \cdot f_{seq}(S_i(t_{ix}, t_{i(x+1)})) \cdot (t_{i(x+1)} - t_{ix})]$. Here, t_{ix} is the starting time of disk i's x^{th} epoch. At the sample moment $t_{i(x+1)}$, we assume there are x workloads running on disk i. $\mathbb{T}_i$ is the set of arrival times of each work running on disk i. The three parts (λ, WAF and time) match the three axes from Fig. 3, where each brink stands for each epoch, and the total volume of these brinks is

the accumulated write count value of that SSD disk. Therefore, TCO' in Eq. 2 can be calibrated as

$$TCO' = \frac{\sum_{i=1}^{N_D} [C_{I_i} + C'_{M_i}(T_{W_i} + \frac{W_i - w_i}{\lambda_i \cdot A_i})]}{\sum_{j=1}^{N_W} \lambda_j (T_{Lf_{D(j)}} - T_{I_j})}. \quad (3)$$

IV. Allocation Algorithms: minTCO

Based on the proposed TCO model, we further design a set of online allocation algorithms, called "minTCO", which adaptively allocate new workloads to SSDs in the disk pool. The main goal is to minimize the data-avg TCO rate (TCO') of the storage pool and meanwhile to conduct disk-lever wear leveling during workload deployment and allocation.

A. Baseline minTCO

The main functionality of the baseline version of minTCO follows these steps: when a new workload arrives, minTCO calculates the data-avg TCO rate for the entire disk pool, and then allocates the workload to the SSD that makes the lowest data-avg TCO rate of the entire dis pool. Specifically, there are two cases during the calculation of expected lifetime and total logical write amount. The first case is that when a new workload is assigned to disk k, we use this new workload's arrival time as the boundary between T_{W_k} and T_{E_k} phases, as shown in Fig. 4(c). The second case is that when the new workload is not assigned to disk k, we use T_{R_k} (the arrival time of the most recent workload on disk k) as the boundary between T_{W_k} and T_{E_k} phases, as shown in Fig. 4(b). As discussed previously, our TCO model is compatible with any WAF models. Here we adopt the WAF model described in Eq. 3 to implement our prototype. The baseline minTCO also needs to consider other resource constraints. For example, minTCO needs to further check if the available spatial and throughput capacities of the chosen SSD are large enough to hold the new workload's working set. If not, minTCO moves to the next SSD which has the second lowest data-avg TCO rate. If no disks have enough capacity, then the workload will be rejected.

B. Performance Enhanced minTCO

One limitation of the baseline minTCO is that it does not balance the load across the SSDs in the pool and thus cannot achieve optimal resources utilization. However, best using of resources (e.g., I/O throughput and disk capacity) is an important goal in real storage system management. To address this limitation, we further develop the performance enhanced manager, namely minTCO-Perf, which considers statistical metrics (i.e., load balancing and resource utilization) as the performance factors in workload allocation.

*1) **System Resource Utilization**:* We consider two types of resources, throughput (IOPS) and space capacity (GB), and calculate the utilization ($U(i,k)$) of disk i when disk k is selected to serve the new workload J_N, as:

$$U(i,k) = \begin{cases} \frac{R_U(i)}{R(i)}, & i \neq k \\ \frac{R_U(i) + R(J_N)}{R_i}, & i = k \end{cases}, \quad (4)$$

where $R_U(i)$, $R(i)$ and $R(J_N)$ represent the amount of *used* resource on disk k, the *total* amount of resource of disk i, and the resource *requirement* of workload J_N, respectively. When i is equal to k, we have the new requirement (i.e., $R(J_N)$) as extra resources needed on that disk. This equation can be used to calculate either throughput utilization (i.e., $U_p(i,k)$) or space capacity utilization (i.e. $U_s(i,k)$). The average utilization can be calculated to represent the system utilization of the entire disk pool: $\overline{U(k)} = \frac{\sum_{k=1}^{N_D} U(i,k)}{N_D}$. Our goal is to increase either average throughput utilization (i.e., $U_p(i,k)$) or average space utilization (i.e. $U_s(i,k)$).

*2) **Load Balancing**:* We use coefficient of variation (CV) of throughput (or space) utilizations among all disks to assess the load balancing. Specifically, when assigning the workload J_N to disk k, we calculate expected $CV(k)$ as: $CV(k) = \frac{\sqrt{\frac{\sum_{i=1}^{N_D} [U(i,k) - \overline{U(k)}]^2}{N_D}}}{\overline{U(k)}}$. A smaller $CV(k)$ indicates better load balancing in the datacenter.

Then, we have our minTCO-Perf algorithm which aims to minimize the data-avg TCO rate, while achieving best resource utilization and load balancing among disks. minTCO-Perf uses an optimization framework to minimize the objective function under constrains listed in Eq. 5.

Minimize:

$$\begin{aligned} & f(R_w) \cdot TCO'(k) \\ & - g_s(R_r) \cdot \overline{U_s(k)} + h_s(R_r) \cdot CV_s(k) \\ & - g_p(R_r) \cdot \overline{U_p(k)} + h_p(R_r) \cdot CV_p(k) \end{aligned}$$

Subject to:

$$\begin{aligned} & i \in D, k \in D \\ & 0 \leq TCO'(i,k) \leq Th_c \\ & 0 \leq U_s(i,k) \leq Th_s \\ & 0 \leq U_p(i,k) \leq Th_p \end{aligned} \quad (5)$$

Upon the arrival of a new workload J_N, we calculate the "enhanced cost" of the the disk pool. The object function in Eq. 5 contains the TCO rate cost ($f(R_w) \cdot TCO'(k)$), the resource utilization reward ($g_s(R_r) \cdot \overline{U_s(k)}$ and $g_p(R_r) \cdot \overline{U_p(k)}$), and the load unbalancing penalty ($h_s(R_r) \cdot CV_s(k)$ and $h_p(R_r) \cdot CV_p(k)$). Notice that $TCO'(k)$ and $TCO'(i,k)$ represent the TCO rate of the entire disk pool and the TCO rate of disk i, respectively, when disk k is selected to take the workload. Non-negative parameters in Eq 5 (e.g., $f(R_w)$, $g_s(R_r)$, $g_p(R_r)$, $h_s(R_r)$ and $h_p(R_r)$) are the weight functions that are related with the read ratio ($R_r = \frac{readIO\#}{totalIO\#}$) and write ratio ($R_w = \frac{writeIO\#}{totalIO\#}$) of workloads. Finally, the disk with the lowest enhanced cost will be selected for the new workload. The reason behind is that in the real world, write intensive workloads affect WAF and TCO, and read intensive workloads are sensitive to load balancing degree. In addition, Th_c, Th_s and Th_p are used as the upper bounds for TCO, space and throughput resources utilization ratios, respectively.

V. Evaluation

A. Write Amplification Measurement and Modeling

*1) **Hardware Testbed**:* Most SSD vendors do not provide APIs or performance counters to measure this physical write quantity. Hence, many prior studies (e.g., [4], [5]) have tried to develop models for *estimating* the WAF of an SSD based on a certain criterion. In this paper, we propose to leverage the data *directly measured* from SSDs to calculate a WAF function for an SSD. Our goal is to characterize the effects of write traffic from multiple workloads on the WAF, and see if such a characterization can be generalized as a mathematical model. Table. I shows the testbed specification. Fig. 5(a) further illustrates the workflow of our measurement experiments.

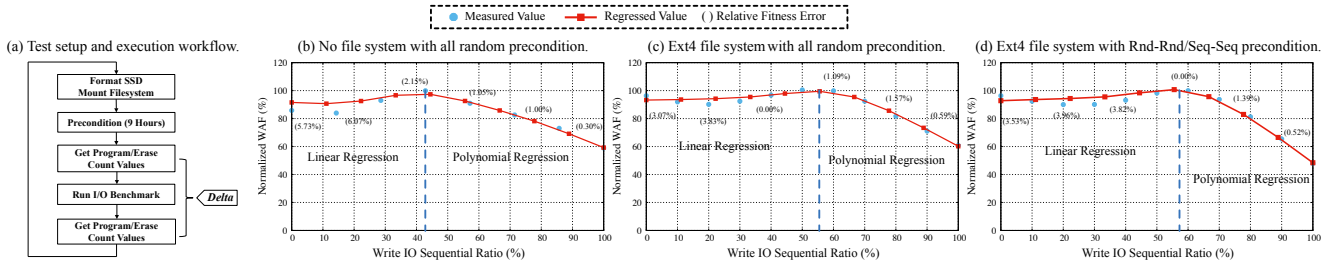

Fig. 5: Test workflow and WAF measurement results.

*2) **Filesystem**:* We test two representative scenarios: "formatting with no file system" and "formatting with Ext4 file system". (1) "No file system" mimics the use case like a swap partition, where avoiding a filesystem mainly has three advantages: making more of the disk usable, since a filesystem always has some bookkeeping overhead; making the disks more easily compatible with other systems (e.g., the *tar* file format is the same on all systems, while filesystems are different on most systems.); and enabling enterprise user to deploy customized block manager running on the hypervisor without mounting a traditional filesystem. (2) "Ext4 file system" is a very solid file system which has been widely used for SSDs in datacenters using linux distributions for the past few years. The journaling that comes with Ext4 is a very important feature for system crash recovery although it causes some acceptable write activity.

TABLE I: Server node configuration.

Component	Specs
Processor	Xeon E5-2690, 2.9GHz
Processor Cores	Dual Socket-8 Cores
Memory Capacity	64 GB ECC DDR3 R-DIMMs
Memory Bandwidth	$102.4GB/s$
RAID Controller	LSI SAS 2008
Network	10 Gigabit Ethernet NIC
Operating system	Ubuntu 12.04.5
Linux Kernel	3.14 Mainline
FIO Version	2.1.10 run with direct I/O
Storage Type	NVMe SSD (Released in 2014)
Storage Capacity	1.6 TB

*3) **Precondition**:* In order to ensure the SSD in the same state and stimulate the drive to the same performance state at the beginning of each measurement, we also conduct a 9 hour "preconditioning process". In detail, we have the following operations: "Sequential precondition" is that between each measurement, the SSD is completely fulfilled with sequentially I/Os, so that all write I/Os in the measurement workloads are overwrite operations, and WAF results will not be independent on garbage collection. "Random precondition" will further conduct an additional completely overwrite to the device with randomly I/Os with 4KB granularity after the sequential precondition process to randomize the workset distribution. "Rnd-Rnd/Seq-Seq precondition" is the policy that we use the random and sequential precondition for non-100% sequential and 100% sequential I/O workloads, respectively. We attempt to use these workloads to observe the ideal write performance (i.e., steady write performance). These two precondition operations can help us simulate different scenarios.

*4) **I/O Workloads**:* In order to study the effects of sequential traffic on WAF, we conduct an experiment that can control the amount of sequential traffic being sent to an SSD. Most workloads in real systems are a certain mixture of sequential and random I/Os. To mimic such real situations, we generate mixed workloads by using an I/O testing tool FIO [9]. We also make changes to an 1.6TB NVMe SSD firmware to parse the value of (page) program and (block) erase counters. We investigate the write amplification factor as the ratio of sequential and random accesses, and the changes of these counters, as shown as "delta" in Fig. 5(a).

*5) **WAF Results and Modeling**:* Fig. 5(b)-(d) show three representative cases from our WAF experimental results, which present the normalized WAF as a function of different sequential ratios on write I/Os. The WAF data points are normalized by the largest WAF across different workload sequential ratios (e.g., the WAF under 40.22% sequential ratio in Fig. 5 (b)). Thus, the original WAF is $\in [1, +\infty)$, while the normalized WAF is $\in [0, 1]$. First, we can see that WAF curves in the three figures are similar, i.e., the curves can be regressed into two stages: a flat linear regression stage and a dramatically decreasing polynomial regression stage. The former part shows that the write amplification factor of mixed workloads is almost identical to that of a pure random workload and keeps almost constant before a turning point (around 40% to 60%). But, after this turning point, the WAF dramatically decreases. In other word, a small fraction of random accesses is necessary to intensify the write amplification factor. We further regress the WAF (A) as a piecewise function of sequentiality of I/O operations in the workload as shown in Eq. 6, where α, β, γ, μ and ε are parameters, and S is the sequential ratio.

$$A = f_{seq}(S) = \begin{cases} \alpha S + \beta, & S \in [0, \varepsilon] \\ \eta S^2 + \mu S + \gamma, & S \in (\varepsilon, 1] \end{cases} \quad (6)$$

At the turning point $S = \varepsilon$, we have $\alpha\varepsilon + \beta = \eta\varepsilon^2 + \mu\varepsilon + \gamma$. Additionally, α is close to zero since the linear regression stage is relatively smooth. We carry out these experiments multiple times on a number of similar devices, and draw our conclusions as follows. We believe that such a mathematical model of sequential write traffic vs. WAF can be constructed for most devices, and each SSD can have its own unique version of WAF function, depending on a number of their hardware-related factors (FTL, wear leveling, over-provisioning etc.). However, being able to regress a mathematical model for the problem forms the basis of the rest of this paper. Additionally, we also observe that the regression turning point of the non-filesystem case (Fig. 5(b)) comes earlier than Ext4's (Fig. 5(c) and (d)). This validates the fact that Ext4's (bookkeeping) overhead is heavier than the raw disk. Moreover, when the sequential ratio is 100%, the WAF under "Rnd-Rnd/Seq-Seq precondition" case (Fig. 5(d)) is lower than that under the "All-Rnd precondition" case (Fig. 5(c)). This validates that in the former case, the steady write performance can be reached.

B. TCO Evaluation

*1) **Benchmarks and Metrics**:* In this section, we plug the WAF model regressed from the real experiments to our TCO

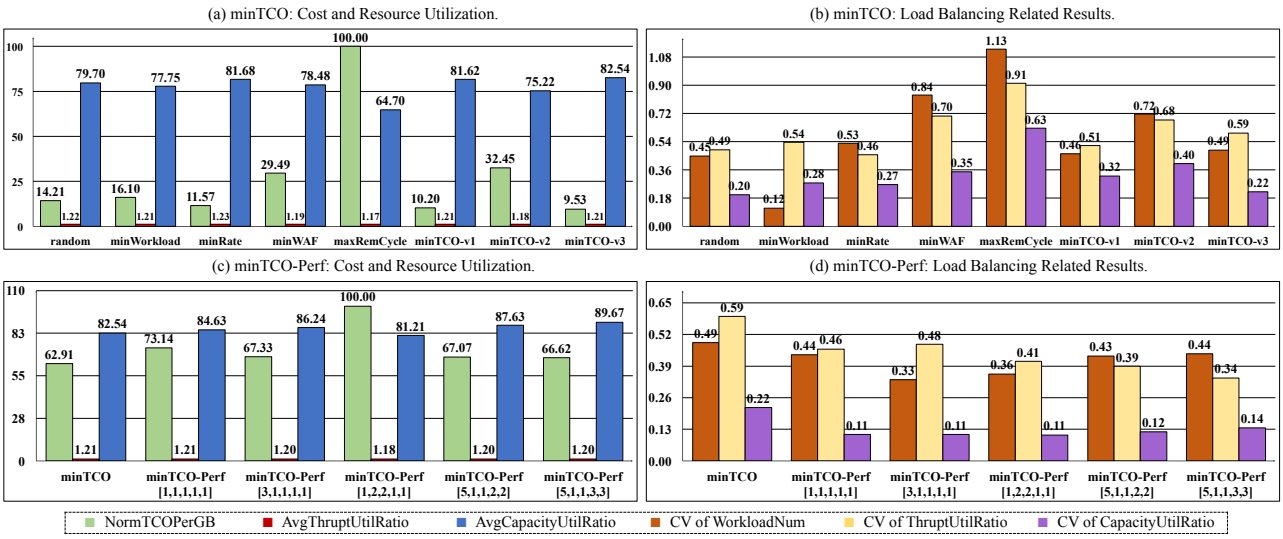

Fig. 6: TCO, resource utilization, and load balancing results under MINTCO and MINTCO-PERF.

model, and then evaluate our MINTCO algorithms. Our trace-driven simulation experiments are conducted based on the spec of real NVMe disks and enterprise I/O workloads. Specifically, we evaluate more than one hundred enterprise workloads from UMass [10], MSR-Cambridge and FIU [11] trace repositories. These workloads represent applications widely used in real cloud storage systems, such as financial applications, web mail servers, search engines, etc.

Table II shows the statistics for some of these workloads (out of more than 100 workloads that we are using), where S is the sequential ratio of write I/O (i.e., the ratio between the amount of sequential write I/Os and the amount of total write I/Os), λ is the daily logical write rate (GB/day), P_{Pk} is the peak throughput demand with the $5min$ statistical analyses window, R_W is the write I/O ratio (i.e., the ratio between the amount of write I/Os and the amount of total I/Os), and WSs is the workings set size (i.e., the spatial capacity demand). The arrival process of these workloads is drawn from an exponential distribution. We use the following metrics to evaluate our MINTCO algorithms: (1) cost per GB during the expected lifetime: the total logical data-averaged TCO during the expected lifetime ($TCO'_{LfPerData}$); (2) resource utilization: the average throughput and space capacity utilization ratios among all disks; and (3) load balancing: the CV of resource utilization ratio across all disks.

TABLE II: Statistics for *part* of I/O workloads we use.

Trace Name	S (%)	λ (GB/day)	P_{Pk} ($IOPS$)	R_W (%)	WSs (GB)
mds0	31.52	21.04	207.02	88.11	6.43
prn0	39.13	131.33	254.55	89.21	32.74
proj3	72.06	7.50	345.52	5.18	14.35
stg0	35.92	43.11	187.01	84.81	13.21
web0	34.56	33.35	249.67	70.12	14.91
hm1	25.15	139.40	298.33	90.45	20.16
hm2	10.20	73.12	77.52	98.53	2.28
hm3	10.21	86.28	76.11	99.86	1.74
onl2	74.41	15.01	292.69	64.25	3.44
Fin1	35.92	575.94	218.59	76.84	1.08
Fin2	24.13	76.60	159.94	17.65	1.11
Web1	7.46	0.95	355.38	0.02	18.37
Web3	69.70	0.18	245.09	0.03	19.21

*2) **TCO Experimental Results**:* We implement both baseline MINTCO and the performance enhanced MINTCO-PERF. Additionally, three versions of MINTCO are considered, such that MINTCO-$v1$ uses the TCO of expected lifetime ($\sum_{i=1}^{N_D}(C_{I_i} + C'_{M_i} \cdot T_{Lf_i})$), MINTCO-$v2$ uses the TCO model of expected lifetime per day ($\frac{\sum_{i=1}^{N_D}(C_{I_i}+C'_{M_i}\cdot T_{Lf_i})}{\sum_{i=1}^{N_D} T_{Lf_i}}$), and MINTCO-$v3$ uses the TCO model of expected lifetime per GB amount ($\frac{\sum_{i=1}^{N_D}(C_{I_i}+C'_{M_i}\cdot T_{Lf_i})}{\sum_{j=1}^{N_W} D_j}$). As expected, none of these baseline MINTCO algorithms consider load balancing and resource utilization during allocation. For comparison, we also implement other widely used allocation algorithms, including $maxRemCycle$ which selects the disk with the greatest number of remaining write cycles, $minWAF$ which chooses the disk with the lowest estimated WAF value *after* adding the newly incoming workload, $minRate$ which chooses the disk with the smallest sum of all its workloads' logical write rates, and $minWorkloadNum$ which selects the disk with the smallest number of workloads.

We first conduct experiments running on the disk pool which consists of 20 disks with 9 different models of NVMe SSDs (available on market in fall 2015). In our implementation, we mix about 100 workloads from MSR, FIU, and Umass with exponentially distributed arrival times in 525 days. Fig. 6(a) and (c) show the results of data-avg TCO rates and resource (I/O throughput and space capacity) utilizations under different allocation algorithms. Fig. 6(b) and (d) further present the performance of load balancing, e.g., CVs of workload number and resource utilizations. First, as shown in Fig. 6(a) and (c), MINTCO-v3 achieves the lowest data-avg TCO rate($/GB). We also observe that among the MINTCO family, MINTCO-$v2$ performs the worst, see Fig. 6(a), and obtains the largest CVs of workload numbers. The reason is that to some extent, MINTCO-$v2$ aims at maximizing the expected life time by sending almost all workloads to a single disk, in order to avoid "damaging" disks increasing the TCO. Therefore, it cannot "evenly" allocate the workloads. We further find that $maxRemCycle$ performs the worst among all allocation algorithms, because it does not consider the TCO as well as the varying WAF due to different sequentialities of the running workloads. In summary, $minTCO$-$v3$ is the best choice which considers expected life time, cost and expected logical data amount that can be written to each disk.

We also implement MINTCO-PERF which is based on

MINTCO-$v3$, and considers the data-avg TCO rate as the criterion to choose the disk for the new workload. As described in Sec. IV-B2, MINTCO-PERF uses Eq. 5 to find the best candidate under the goal of minimizing TCO, maximizing resource utilization, and balancing the load. There are a set of weight functions (i.e., $f(R_w)$, $g_s(R_r)$, $g_p(R_r)$, $h_s(R_r)$ and $h_p(R_r)$) used in Eq. 5 . To investigate the effects of these weight functions, we conduct sensitivity analysis on the average values for the five weight functions in Eq. 5. After trying different approaches, and choose the linear function approach to implement weight functions. We show the over-time average value of each function normalized by the minimum function one. For example, "[5,1,1,2,2]" means that all values are normalized by the second weight function ($g_s(R_r)$). We also observe that space capacity (instead of I/O throughput) is always the system bottleneck (i.e., with high utilization) across different approaches. This is because NVMe SSDs support up to $64K$ I/O queues and up to $64K$ commands per queue (i.e., an aggregation of $2K$ MSI-X interrupts). Meanwhile, workloads we are using here are collected from traditional enterprise servers, which have not been optimized for NVMe's revolutionary throughput capacity improvement. We also find that with a slight sacrifice in TCO, MINTCO-PERF can improve both resource utilization and load balancing. Fig. 6(c) further show that "[5,1,1,3,3]" is the best choice among all cases, which is 3.71% more expensive than the baseline $minTCO$, but increases the space utilization ratio by 7.13%, and reduces CV of throughput and space capacity utilization by 0.25 and 0.8, respectively. This is because MINTCO-PERF sets TCO and space capacity higher priorities.

VI. RELATED WORK

Few prior studies that have focused on the long-term costs of SSD-intensive storage systems with SSDs so far, especially in the context of datacenters. Majority of the existing literature investigates SSD-HDD tiering storage systems. [12] was proposed to reduce the cost of a SSD-HDD tiering storage system by increasing both temporal and spatial update granularities. The "cost" in vFRM is the I/O latency and delay, rather than price. [13] built a cost model that also considers the lifetime cost of ownership including energy and power costs, replacement cost, and more. They assume that the "trade-in" value of the disk is a linear function of its available write cycles. Meanwhile, in terms of budget-driven workload allocation method, [14] recently presents a systematic way to determine the optimal cache configuration given a fixed budget based on frequencies of read and write requests to individual data items. Write amplification of an SSD depends on the FTL algorithm (e.g., [15], [16]) deployed in the controller. However, SSD vendors do not publicly reveal the FTL algorithms to customers due to confidentiality issues. Techniques for isolating random and sequential I/Os and VM migration I/O allocation were presented in [17] and [18]. To prolong the lifetime and improve the performance of SSDs with cache, [19] proposed an effective wear-leveling algorithm based on a novel "triple-pool" design. [4] presented a nearly-exact closed-form solution for write amplification under greedy cleaning for uniformly-distributed random traffic.

VII. CONCLUSION

In this paper, we characterize the write amplification (WA) of SSDs as a function of fraction of sequential writes in a workload. We plug this WAF function into our proposed Total Cost of Ownership (TCO) model, which also considers capital and operational cost, estimated lifetime of Flash under different workloads, resource restrictions and performance QoS. Based on the TCO model, we build the online workload allocation algorithm MINTCO and MINTCO-PERF. Experimental results show that MINTCO reduces the ownership cost, and MINTCO-PERF further balances the load among disks and maximize the overall resource utilization, while keeping the TCO as low as possible.

REFERENCES

[1] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang *et al.*, "Bigdatabench: A big data benchmark suite from internet services," in *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*. IEEE, 2014, pp. 488–499.

[2] "Amazon Web Services," http://aws.amazon.com.

[3] "Google Computer Engine," http://cloud.google.com/compute.

[4] P. Desnoyers, "Analytic modeling of ssd write performance," in *Proceedings of the 5th Annual International Systems and Storage Conference*. ACM, 2012, p. 12.

[5] W. Bux and I. Iliadis, "Performance of greedy garbage collection in flash-based solid-state drives," *Performance Evaluation*, vol. 67, no. 11, pp. 1172–1186, 2010.

[6] R. Zhou, M. Liu, and T. Li, "Characterizing the efficiency of data deduplication for big data storage management," in *Workload Characterization (IISWC), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 98–108.

[7] V. Tarasov, D. Hildebrand, G. Kuenning, and E. Zadok, "Virtual machine workloads: The case for new nas benchmarks," in *Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13)*, 2013, pp. 307–320.

[8] "The mega-datacenter battle update: Internet giants increased capex in 2014," http://451research.com/report-short?entityId=84745&referrer=marketing.

[9] "FIO: Flexible I/O Tester," http://linux.die.net/man/1/fio.

[10] "UMass Trace Repository," http://traces.cs.umass.edu.

[11] "SNIA Iotta Repository," http://iotta.snia.org/historical_section.

[12] J. Tai, D. Liu, Z. Yang, X. Zhu, J. Lo, and N. Mi, "Improving flash resource utilization at minimal management cost in virtualized flash-based storage systems," *IEEE Transactions on Cloud Computing*, vol. 7, no. 99, p. 1, 2015.

[13] Z. Li, A. Mukker, and E. Zadok, "On the importance of evaluating storage systems' costs," in *Proceedings of the 6th USENIX Conference on Hot Topics in Storage and File Systems (HotStorage '14)*, 2014, pp. 1–5.

[14] S. Ghandeharizadeh, S. Irani, and J. Lam, "Memory hierarchy design for caching middleware in the age of nvm," 2015.

[15] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, and H.-J. Song, "A log buffer-based flash translation layer using fully-associative sector translation," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 3, p. 18, 2007.

[16] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A space-efficient flash translation layer for compactflash systems," *Consumer Electronics, IEEE Transactions on*, vol. 48, no. 2, pp. 366–375, 2002.

[17] A. Povzner, T. Kaldewey, S. Brandt, R. Golding, T. M. Wong, and C. Maltzahn, "Efficient guaranteed disk request scheduling with fahrrad," in *ACM SIGOPS Operating Systems Review*, vol. 42, no. 4. ACM, 2008, pp. 13–25.

[18] J. Roemer, M. Groman, Z. Yang, Y. Wang, C. C. Tan, and N. Mi, "Improving virtual machine migration via deduplication," in *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2014, pp. 702–707.

[19] W.-n. . N. Wang, F.-s. . S. Chen, and Z.-q. . Q. Wang, "An endurance solution for solid state drives with cache," *Journal of Systems and Software*, vol. 85, no. 11, pp. 2553–2558, 2012.