Latency/Wearout in a Flash-based Storage System with Replication on Write

Alexander Rumyantsev, Evgeny Ivashko, Ilya Chernov
Institute of Applied Mathematical Research, KRC of RAS
Petrozavodsk State University
Petrozavodsk, Russia
{ar0, ivashko, chernov}@krc.karelia.ru

Dmitry Kositsyn, Anton Shabaev, Vadim Ponomarev
Petrozavodsk State University
Petrozavodsk, Russia
kositsyn@psu.karelia.ru, ashabaev@petrsu.ru, vadim@cs.karelia.ru

***Abstract*—We investigate the influence of latency and wearout of writing data on the efficiency of high-performance data storage systems. Using a mathematical model, we propose the strategy of redundant writing with cancelling long write operations. Presented numerical simulation results allow choosing the replication level needed to achieve the trade-off between the latency and wearout that provides desired reliability and efficiency.**

I. Introduction

Life today is hardly possible without information technologies. They are used everywhere, from nuclear power stations to text messaging between people. In many applications, high speed information service is of critical importance. This can include intensive input-output of data, high peak load, etc. Examples are search engines like Yandex or Google, social networks, file exchange networks, video on demand, etc. Such systems are usually heavily loaded. They need special technologies, in particular, for data storage. Data storage subsystems of high-performance services must provide high data availability and quick reading and writing. However, this implies using special strategies of data duplication (replication).

Recently, since the development of the flash-based storage systems, solid state drives (SSD) are widely used due to higher efficiency and, in particular, sequential reading speed. This property promotes the usage of SSD in high-performance systems that possess intensive read and write data flows. Replication of data among several data storage systems is able to reduce the response time, provide data integrity in case of failures, however suffering from higher wearout rate of SSD devices. At that, it is necessary to obtain a replication strategy that balances the response time reduction with the wearout.

The contribution of this paper is twofold. First, we adopt a strategy of data replication and quorum of completed operations to reduce write latency. This strategy relies on SSD's feature of cancelling a write operation in progress. The main idea of such a strategy is to replicate the write operations in order to accept the fastest ones and cancel, using the write cancellation mechanism, the slower ones. This promises improvement of the response time of the whole system. We further develop the so-called multiserver Split-Merge model to obtain the necessary analytical results for obtaining the configuration of a system delivering minimal wearout. In particular, we state the non-linear equation to obtain the minimal wearout of storage devices for given system configuration, for the first time.

Second, we present results of numerical experiments to find the balance between the acceptable latency and wearout rate. As the experiments show, one can use the proposed technique of redundant write with cancellation, provided a sufficient amount of devices, to reduce latency, while the lifetime of the devices is affected in a minor way. We focus on the distributions of request duration with so-called heavy tails which is related to the modern storage characteristics.

The structure of the paper is as follows. We state the problem of replication and quorum strategy optimization in section II. In section III we give the necessary information about the SSD-devices. The related works are surveyed in section IV. The mathematical model is developed in section V, and the results of numerical experiments are presented. Finally, in section VI we sum up the results and give a conclusion.

II. Problem Statement

In this paper, we propose a strategy that employs both replication and the quorum technique. Replication is used to provide the necessary efficiency: when a write request arrives, the piece of data is written concurrently at several carriers. The quorum approach is used as follows: the write request is said to be completed, when the write operation is completed at a certain number of carriers called the quorum, which is not greater than the number of devices that received the write request (the replication). Such information redundancy among several carriers allows, besides data integrity, to obtain a higher reading speed: during the read operations the data is read from the quickest carrier, or pieces of data are read from a few carriers in parallel.

Such approach is based on data storage redundancy and redundancy of writing operations. Redundancy has its cost, including the necessity to have a supply of carriers, higher hardware wear and energy consumption, etc. Therefore, we face a trade-off between the positive effect and the necessary cost of using quorum and replication. In the paper we obtain optimal configuration of replication and quorum by means of mathematical modeling.

To analyze the effect of latency and wearout on the write performance of an SSD-based storage system, we further develop the multiserver Split-Merge-type model first proposed in [1], where a study of latency reduction by means of replication in Desktop Grid environment was performed. A further study of the effects of tail distribution of service times

ISSN 2305-7254

and structure of the arrival process in Split-Merge model by means of simulation was performed in [2]. In [3] a necessary background on the order statistics of heavy-tailed service time distributions and properties of linear (in replication parameter) cost functions in the Split-Merge model was established. In the present paper we further develop the Split-Merge model in the context of high-performance SSD-based storage systems.

III. Technical Details

Solid-state data-storage drives based on the flash technology are quickly developing and, as permanent storage devices, are an alternative to hard disk drives. Due to flash's energy efficiency, quick read access, compact sizes, resistance to shocks and vibrations, relatively high reliability they are of high interest as storage drives in servers and data-storage systems.

Drawbacks of SSD, compared to HDD, include: higher cost per 1Tb (though this difference is rapidly decreasing) and relatively quick wearout of SSDs, up to complete failure of the device.

The architecture of SSD (see fig. 1) includes the flash-memory units, the embedded processor unit equipped by its own RAM, and a set of controllers to provide interaction between these components.

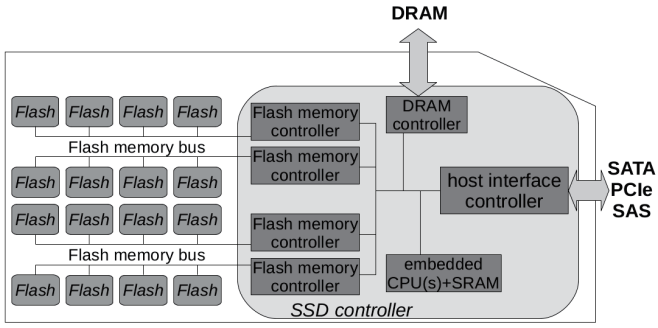

Fig. 1. General architecture of an SSD

Each SSD has special built-in software called the flash translation layer (FTL). It is usually implemented as the internal software (firmware) launched by the SSD's controller. This controller transforms incoming read/write requests to the flash-memory operations via the flash-controller. The host interface connects the disk with the host via the interface connector, e.g., SATA. Also, an SSD has a RAM buffer, which is used for temporarily storing data and buffering write requests.

Data are permanently stored in the array of flash-memory cells that are connected with the flash-controller via several channels. Each flash-memory cell consists of a few crystals, each of which has several matrices. Every matrix contains more than one flash blocks. See, e.g., [4] for a more detailed description of the SSD architecture.

Built-in memory and processor allow to create a queue of read/write requests to SSD; a similar queue can exist also outside the SSD.

Data stored in an SSD are accessed up to a page of a fixed size; such pages can be considered as SSD blocks. The difference is that a page can not be locally updated, only rewritten completely after a complete clearing. A clearance block consists of more than one (usually 64 or 128) consequent memory pages; so, all contents of these pages must be copied prior to erasing the block. This means a serious problem of written volume increase for SSD.

To veil this behaviour, SSDs use flash transition layers (FTL). FTL maps logical page numbers onto physical page numbers, creating an illusion of local updates. Another important function of the FTL is garbage collection. It is clearing one or more blocks when there is a lack of free pages to serve write requests or when the drive is idle. Usually a block with the fewest number of up-to-date data on its pages is chosen for garbage-collection, in order to minimize redundant writing to the drive. Besides, as SSD support a limited number of erase cycles, FTL levels out the wearout by distributing write operations evenly along the blocks and, therefore, increases the lifetime of the drive.

Besides, parallel serving of requests may be impossible if two or more requests address the same flash chip. Sequential service means latency, because otherwise the requests could be fulfilled simultaneously. Reading can be 10 to 40 times faster than writing, causing another source of latency [5].

As writing is subject to multiple factors that influence the writing rate, it can vary significantly. See [5] for a more detailed description of read-write characteristics of SSD.

The number of erases for each block is limited by a number of erases from 10 thousand to 1 million; this restricts the lifetime of SSD [6]. So, write operations must be optimized in order to keep the drive running for a long time. Usually the drive's reliability is measured as the amount of information that can be written without loss of the device's capacity.

Also, there is an undesirable phenomenon called the write amplification effect, when more information is actually written compared to the amount requested to be written. It is typical for flash memory and SSDs.

In this work we propose a replication strategy for write requests for several parallel SSD. The request is fulfilled when the given number (the quorum) of replicated write operations are completed. Other operations, still not completed, are cancelled. The possibility of cancelling running write operations on SSD is described in [7].

IV. Related Work

High-performance storage is a storage management system especially designed for moving large files, large amounts of data and process high I/O load around a network.

It is quite important to take into account drive-type specifics when designing storage systems for high-performance hardware, and this importance is increasing. Disk-oriented solutions are not easily scalable and thus not always meet the needs of large scale applications, e.g., the Web ones. Drive capacity has been rapidly growing while progress in reducing latency and increasing bandwidth is not so drastic.

For this reason, high IO-load storage systems are built on DRAM. This storage type provides the best performance for market solutions. For example, the authors of [8] propose

the RAMClouds solution, which, according to the authors, can provide durable and available storage with 100-1000x the throughput of disk-based systems and 100–1000 times lower access latency. However, DRAM storage suffers some drawbacks, including high cost and energy consumption per bit. From these points of view, the RAMClouds storage, as admitted by the authors, can be 50–100 times worse compared to a HHD-based storage. Besides, a DRAM storage system demands more room in a datacentre compared to HDD or SSD storage units, due to lower data density.

Quick development and expansion of SSD is a reason to design SSD-based storage systems, or at least those where SSD play an important role. Such storages can have different architectures, beginning from the all-flash storages made of SSD only, up to hierarchical storages, where the SSG-subsystem is between the quick DRAM-based cache and relatively slow permanent HDD-based storage and is both a slow cache and fast long-term storage.

One of the first solutions of this type was the high performance hybrid storage system, called Hystor [9]. Hystor monitors I/O access patterns at runtime and, therefore, is able to reveal effectively those blocks that either threat long latencies or are semantically critical (e.g., file system metadata). Such blocks are then stored at SSD: this promises a significant gain in performance. Also, Hystor serves as a write-back buffer in order to improve the speed of write requests by applying exceptionally high performance of writes in the state-of-the-art SSDs,

The solution proposed in this paper can be possibly used both in an all-flash storage and in high-performance caches of hierarchical storages.

Increasing efficiency of the SSD layer or of the all-flash storage under high IO-load is attracting attention of many researchers. Different levels of the storage can be optimized.

Modern SSDs provide a high parallelizm inside a drive (see details in [5]). SSD controllers are designed to exploit the hardware parallelism as completely as possible. In particular, some sub-requests can wait significantly longer than other, if they are aimed at different flash chips, because queues at chips are not necessarily the same, also loads are not always balanced. A request is fulfilled only when all sub-requests are, so some sub-requests need to wait idle. The authors of [5] propose a new class of schedulers that are able to leverage such sub-request disbalance and, thus, to shorten the response time. The scheduler named Slacker is a slack-enabled re-ordering scheduler; its design and implementation are presented in the paper. Slacker is layered under the modern SSD request scheduler; it estimates the slack of each incoming sub-request to a flash chip and may place them ahead of existing sub-requests with sufficient slack. This reduces the detrimental impact on the response time. Slacker's implementation is quite simple and demands only slight additions to the hardware. The authors have tested Slacker on 21 workloads with different read-write characteristics, and show that Slacker provides 19.5%, 13%, and 14.5% improvement in response time; the with average improvement is 12%, 6.5%, and 8.5%, for write-intensive, read-intensive, and read-write balanced workloads, respectively.

An attempt to increase the lifetime and performance of secure memory is done in [10]. DeWrite is presented; it deduplicates writes in-line and judiciously integrates deduplication and encryption to deliver high performance. DeWrite is based on existing metadata store and metadata cache of secure disks and adds only deduplication logic into the memory controller, achieving the low design complexity. Experiments show that DeWrite eliminates 54% of writes, and speeds up memory writes and reads by 4.2× and 3.1×, on average. Meanwhile, DeWrite improves the IPC by 82% and reduces 40% of energy consumption on average.

So, on-the-fly deduplication is able to improve SSD use efficiency. However, deduplication does not grant a significant effect on every load type. See [11] for a detailed survey of research in the area of deduplication on SSD. In this work, we propose an inverse approach: to use redundant writes for higher performance of a group of SSD drives.

Flash possesses an interesting mechanism of cancelling or suspending write operations. It can be used for building different strategies of scheduling such operations in order to increase the SSD performance.

Write and read latencies of a flash device are different, with the former much higher than the latter. In NAND flash memory, read requests are suspended to wait until the time-consuming page program or block erase (P/E) operation is completed. According to the preliminary results, P/E operations increase the read latency by two on average; obviously, this may spoil the total performance of the system. In [12], the authors propose a low-overhead P/E suspension scheme, basing on the internal mechanism of NAND flash P/E algorithms. The P/E may be suspended to service pending reads and then resumed. So reads have the top priority; the approach is further extended by allowing writes to preempt erase operations for decreasing the write latency. Results of experiments are reported: a realistic SSD simulation model that adopts multi-chip/channel and evaluate NAND flash as storage materials of diverse performance was run to show the near-to-optimal performance of read requests provided by the proposed technique; the write latency is also significantly reduced. on average, the reduction was 46.5% compared to the RPS (Read Priority Scheduling); when using write-suspend-erase, the write latency is reduced by 13.6% relative to FIFO.

Besides, higher write latency can be reduced using buffers. However, write requests scheduled to a memory bank can increase latency for subsequent read requests to the same bank. Write requests are shown to increase the effective read latency 2.3 times on average, causing significant performance degradation, in [7], where the baseline flash system with read-priority scheduling is studied. To overcome this degradation, i.e., to reduce the read latency of flash devices under such scenarios, the authors propose adaptive Write Cancellation policies: processing of a scheduled write request can be aborted if a read request arrives at the same bank within a predetermined period of time. Also Write Pausing, which uses the iterative write algorithms of an SSD to pause at the end of each write iteration in order to service any pending reads is proposed. For baseline systems, the proposed technique is shown to remove 75% of the latency increase incurred by read requests and improves overall system performance by 46% on average. Necessary hardware changes are negligible, extensions in the SSD controller are rather simple.

V. Mathematical Model

In what follows we use the term *server* to indicate a single SSD serving write request received by n-server storage *system*. Each such write request, called *customer* below, induces r write operations to distinct SSDs (*tasks*). The number of tasks per customer is known as the *replication* factor, r. To guarantee consistency (and possibly to accelerate future read operations), we require q of r tasks per customer to be successful, in this case the system works with *quorum* q.

To keep analytical tractability, we make the following model assumptions:

- task durations are independent and identically distributed (iid.) random variables with the (general) distribution function F;
- all $r \leqslant n$ tasks of a customer are started simultaneously;
- right after successful termination of $q \leqslant r$ tasks of a customer, redundant $r - q$ tasks are immediately cancelled;
- customers are waiting in a single unbounded First-Come-First-Served queue.

These assumptions allow to treat the model as the classical $\lfloor n/r \rfloor$-server model with specific service time distributions. More precisely, the service time of a customer is the q-th order statistics of r iid. service times of tasks. Following [3], we use the notation $S_{q:r}$ to indicate the general service time of a customer (for fixed q,r) distributed as $F_{q:r}$, where

$$F_{q:r}(x) = \sum_{i=q}^{r} \binom{r}{i} F^i(x)(1 - F(x))^{r-i}, \qquad (1)$$

(see, e.g., [13]).

Now we formally define the performance measures studied further in the paper:

Latency $l(q,r) := \mathrm{E}D + \mathrm{E}S_{q:r}$, where $\mathrm{E}D$ is the mean stationary delay of a customer,

Wearout $w(q,r) := r\mathrm{E}S_{q:r}$ is related to the amount of work (the volume of performed write) done per customer.

It now follows from the well-known stability condition of the multiserver model, that to guarantee $l(q,r) < \infty$, the arrival intensity λ of customers must satisfy the following inequality:

$$\lambda < \lambda^*(q,r) := \left\lfloor \frac{n}{r\mathrm{E}S_{q:r}} \right\rfloor = \left\lfloor \frac{n}{w(q,r)} \right\rfloor. \qquad (2)$$

Thus, $\lambda^*(q,r)$ is the maximal throughput of the system for given q, r. Note, also, that the maximal throughput is inversely proportional to wearout.

A. Wearout Analysis

The wearout $w(q,r)$ is a function of two parameters in discrete parameter space $1 \leqslant q \leqslant r \leqslant n$. Thus, numerical methods can be used to study $w(q,r)$ for relatively small values of n. However, some analytical results simplify the analysis and allow to gain deeper insight into the model properties. Most of the results follow from the properties of order statistics [14]. In particular, it can be shown [3] that

$$w(q,r) \geqslant w(q-1,r), \qquad q > 1, \qquad (3)$$
$$w(q,r) \geqslant w(q-1,r-1), \; q,r > 1. \qquad (4)$$

That is, the wearout $w(q,r)$ is non-decreasing with *increasing quorum* q (for fixed replication r), as well as *increasing quorum* q *and replication* r simultaneously. However, it remains unclear how increasing the replication (for fixed quorum) infers the wearout, which is related to the following equality [14]:

$$F_{q:r}(x) = F_{q:r-1}(x) + \binom{r-1}{q-1} F^q(x)\overline{F}^{r-q}(x), \quad q \geqslant 1, \; (5)$$

where $\overline{F}(x) := 1 - F(x)$ is the tail of the distribution. In particular, this means that $\mathrm{E}S_{q:r} \leqslant \mathrm{E}S_{q:r-1}$, which motivates the need to study $w(q,r)$ as a function of r (for fixed q).

Due to properties (3)–(4), it is important to study the function $w(1,r)$, i.e., the wearout for fixed $q = 1$. The following result proved in [15, Lemma 1] relates the properties of $w(1,r)$ and the so-called log-concavity/convexity of the service time distribution:

Lemma 1. *[15] If $\overline{F}(x)$ is log-concave (log-convex), then $w(1,r)$ is non-decreasing (non-increasing) in r.*

The *log-concavity* [16], [17] is an important property related to the well-known new-better-than-used (NBU) and increasing failure rate (IFR) classes of distributions widely used in reliability analysis. Uniform, normal, logistic, Weibull (with shape $\geqslant 1$) distributions are log-concave, while Weibull (with shape < 1) is log-convex, to name a few. Exponential distribution is log-concave and log-convex simultaneously, that induces *independence* of $w(1,r)$ on r for exponentially-distributed service time S. Indeed, it is easy to see that $w(1,r) = \mathrm{E}S$, since $\mathrm{E}S_{1:r} = \mathrm{E}S/r$. However, there are distributions that are neither log-concave, nor log-convex, and in such a case numerical study is unavoidable. We also note that the proof of Lemma 1 relies on (1) for $q = 1$, and can not be generalized to arbitrary $q > 1$ easily. Moreover, internally it assumes that $F(x)$ is defined on $[0, \infty)$ which is not valid for Pareto distribution (although formally Pareto is log-convex). Thus, in what follows we study the $w(q,r)$ for special classes of distributions, focusing on the distributions with heavy tails.

The evidence of heavy tails in distributions of files stored at the user/server level has been reported in many studies, see, e.g., [18], [19], [20]. Such a class of distributions possesses many properties that complicate the analysis or dramatically infer the system performance [19]. In particular, a heavy-tailed random variable may have large coefficient of variation, which is mostly caused by the so-called mass-count disparity. Intuitively this means that large number of smaller items occupy only a small amount of storage, whereas the majority is occupied by several largest items. To illustrate this, the so-called mass-count disparity plot suggested in [18] can be used. The mass-count disparity of system and user file sizes distribution on a contemporary SSD storage is depicted in Fig. 2. The so-called Pareto rule for this sample is 5/95, which means that 95% of files totally occupy only 5% of the storage. Thus, we perform the following analysis for the heavy-tailed

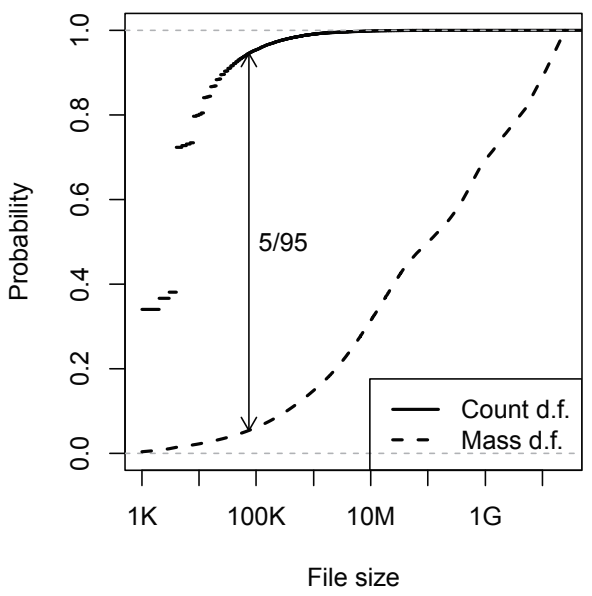

Fig. 2. Mass-count disparity of the UNIX file sizes in a storage. The count d.f. (e.c.d.f.) and mass d.f. (empirical integrated tail d.f.) vs. file size (logarithmic scale). The joint ratio indicated by the arrow is 5/95 (5% of storage occupied by 95% of files)

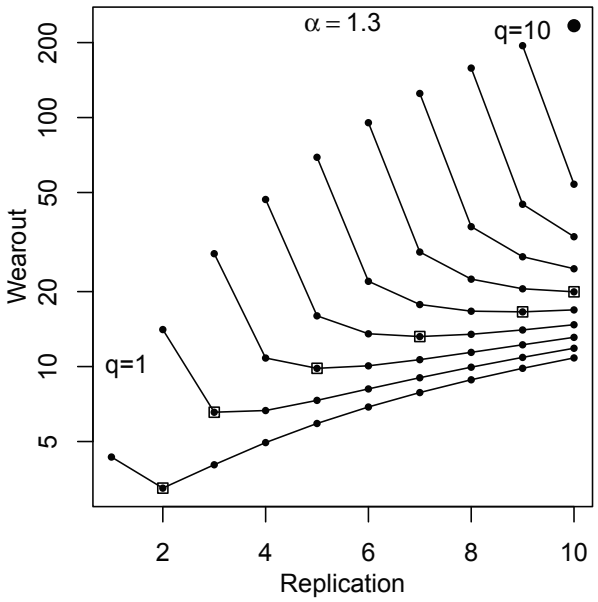

Fig. 3. $w(q,r)$ vs. r for a given $q = 1,\dots,10$ for Pareto distribution of service times with $\alpha = 1$ .3. Minimal wearout $w(q,\ r^*(q))$ are highlighted with squares, where $r^*(q)$ is defined in (7)

class, focusing on widely adopted standard Pareto and Weibull distributions.

Now we discuss the properties of $w(q,r)$ for Pareto and Weibull service time distributions. Recall the standard one-parameter Pareto distribution function

$$\mathrm{P}(S \leqslant X) = 1 - x^{-\alpha}, \quad x \geqslant 1.$$

It can be seen that $\mathrm{E}S^k < \infty$ for $k < \alpha$, that is, Pareto-distributed service times may have infinite moments (mean, variance) for specific values of α. Moreover, it can be clearly seen that the tail of Pareto distribution is log-convex [17]; however, the support of standard Pareto distribution is $(1,\infty)$, and thus Lemma 1 is not applicable. At the same time, it can be deduced that [3]

$$w(1,r) = r + \frac{1}{\alpha} + \frac{1}{\alpha(\alpha r - 1)}, \quad r > 1/\alpha. \tag{6}$$

In particular, (6) means that $w(1,1)$ may become large if α is close to 1. Indeed, $w(1,1) = \mathrm{E}S = \alpha/(\alpha-1)$. Note that a Pareto random variable with $\alpha < 2$ has infinite variance. At the same time, $w(1,2) < 2 + 2/\alpha$, since $\alpha r - 1 > 1$ for $r \geqslant 2$. That means that $w(1,2)$ can be significantly smaller than $w(1,1)$. However, it can be seen that $w(1,r)$ increases in r for $r > 2$.

Moreover, it can be shown [3] that $w(q,r)$ non-decreases in r for $r \geqslant r^*(q)$ defined as

$$r^*(q) := \left\lceil \frac{1 + q + \alpha q - 2\alpha + \sqrt{(\alpha q + q - 1)^2 + 4q}}{2\alpha} \right\rceil, \tag{7}$$

and non-increases in r for $r < r^*(q)$, thus $r^*(q)$ attains minimal wearout $w(q, r^*(q))$ for a given q. To illustrate these results, the dependence of $w(q,r)$ on r for fixed q is depicted in Fig. 3

Standard Weibull distribution is defined as follows:

$$\mathrm{P}(S \leqslant X) = 1 - e^{-x^{\xi}}, \quad \xi > 0.$$

It can be shown that Weibull distribution is log-concave for $\xi > 1$ and log-convex for $\xi < 1$ [16], while it reduces to the standard exponential distribution for $\xi = 1$. Thus, Lemma 1 is applicable and $w(1,r)$ is non-decreasing (non-increasing) in r for $\xi > 1$ ($\xi < 1$).

However, the dependence of $w(q,r)$ on r for general q remains unclear. Note that for Weibull service time distribution

$$w(q,r) = r\frac{r!}{(r-q)!}\frac{\Gamma(\xi^{-1})}{\xi}\sum_{j=0}^{q-1}(-1)^j\frac{(r-q+1+j)^{-1-\xi^{-1}}}{j!(q-1-j)!}.$$

Straightforward analysis of the difference $w(q,r) - w(q,r+1)$ induces that $w(q,r)$ is non-decreasing if the following sum is non-negative:

$$\phi(q,r) := \sum_{j=0}^{q-1}\binom{q-1}{j}\left[\frac{r^2}{(r-q+1+j)^{1+\xi^{-1}}} - \right.$$
$$\left. - \frac{(r-1)(r-q)}{(r-q+j)^{1+\xi^{-1}}}\right].$$

and is non-increasing otherwise. As preliminary numerical experiments show, for $\xi > 1$ (the heavy-tailed Weibull distribution) and fixed q there exists a single solution of the equation $\phi(q,x) = 0$, such that $r^*(q) = \lfloor x \rfloor$ delivers the minimal wearout for a given q. We depict $w(q,r)$ as a function of $r \leqslant 10$ for fixed $q \leqslant r$ on Fig. 4.

Finally, note that the expression $\phi(q,r)$ is slightly simplified for $\xi = 1$, when the service times are exponentially

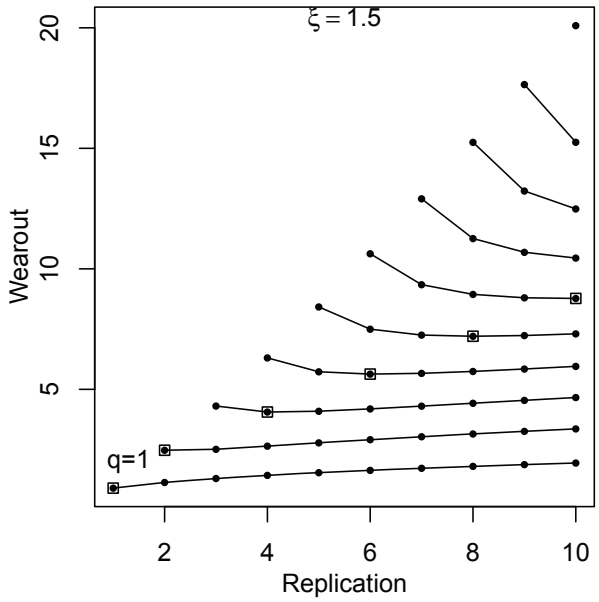

Fig. 4. $w(q,r)$ vs. r for a given $q = 1,\dots,10$ for Weibull distribution of service times with $\xi = 1.5$. Minimal wearout $w(q, r^*(q))$ is highlighted with squares, where $r^*(q)$ is determined by the solution of the nonlinear equation $\phi(q,\, x) = 0$

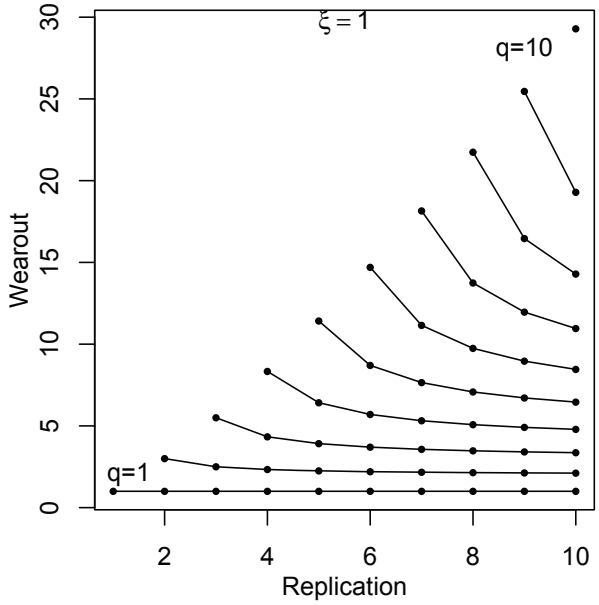

Fig. 5. $w(q,\ r)$ vs. r for a given $q = 1\,,\dots,10$ for exponential distribution of service times. Note that the wearout is monotone non-increasing in r

distributed; numerical experiments show that $\phi(q,r) < 0$ for all $r \geqslant q > 1$. However, we leave the rigorous analysis of this phenomenon for future research. To conclude this section, we illustrate the $w(q,r)$ for exponentially distributed service times in Fig. 5, where the independence of $w(1,r)$ on r, as well as non-increasing $w(q,r)$ for $q \geqslant 1$, is clearly seen.

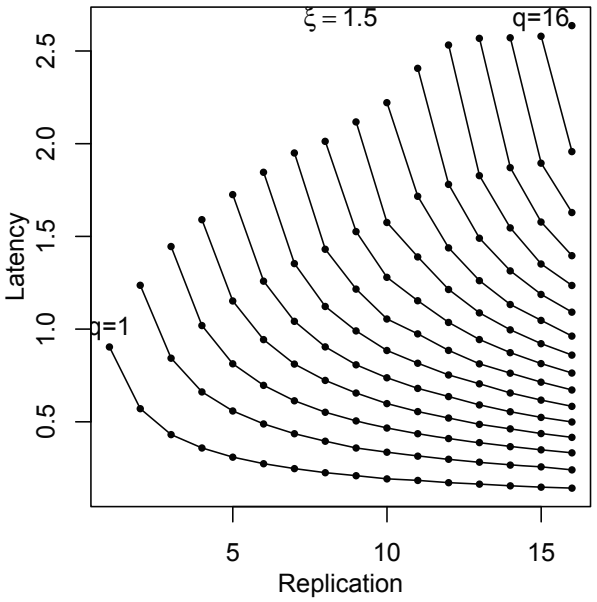

Fig. 6. $l(q,\ r)$ vs. r for given $q = 1\ ,\dots,16$ for Weibull distribution of service times with $\xi = 1\ .5$

B. Latency Analysis

Since the closed form expression is available only for less interesting case of $M/G/1$-type system (some explicit results for this case can be found in [3]), we use simulation to study the dependence of latency $l(q,r)$ on the parameters q, r. We use the package `queuecomputer` for the R language to simulate the multiserver system and obtain latency estimates. We set $n = 32$ and vary $1 \leqslant q \leqslant r \leqslant 16$ to obtain $l(q,r)$. We illustrate the results of simulation for Weibull service time distribution with $\xi = 1.5$ and exponentially distributed inter-arrival times, setting the arrival rate for each fixed r in such a way that $\lambda \mathrm{E} S_{r:r} = 0.5\lfloor n/r \rfloor$ to guarantee stability for the largest $q = r$. It can be seen from Fig. 6 that the latency is monotone with respect to r for a given q.

The results of experiments for Pareto distributed service times with $\alpha = 1.5$, ceteris paribus, exhibit a similar pattern. Note, however, that the latency dramatically increases: this is caused by huge variation of service times (Fig. 7).

To sum up the numerical observations, note that both for Pareto and heavy-tailed Weibull distributions, the latency decreases with r, whereas the wearout finally increases with r; this leaves a possibility to obtain the necessary latency/wearout tradeoff. At the same time, a conservative approach of minimizing the wearout allows increasing the throughput $\lambda^*(q,r)$, possibly sacrificing the latency.

VI. Conclusion

In this paper we present a strategy of replicating write requests in high-performance SSD-storage systems. The strategy is based on a mathematical model of the M/G/1 multi-server system and considers incoming request flow as Pareto or Weibull distributed; this agrees with modern research of internet services load. The proposed strategy allows choosing an effective latency/wearout ratio to provide the necessary

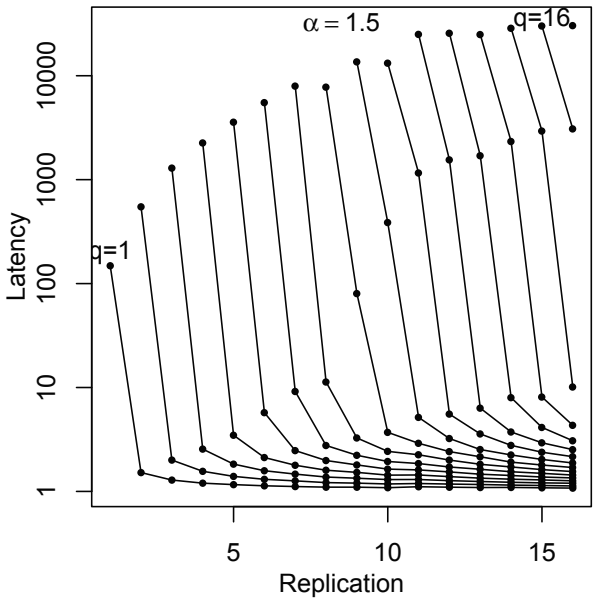

Fig. 7. $l(q,\ r)$ vs. r for given $q = 1\ , \ldots , 16$ for Pareto distribution of service times with $\alpha = 1\ .5$; note the logarithmic scale for the latency

performance at the lowest cost. Simulation results illustrate the advantages of the strategy. The future work is to implement the strategy and testing it in real load conditions.

ACKNOWLEDGMENT

This work is supported by the Ministry of Education and Science of Russian Federation [project no. 14.580.21.0009, unique identifier RFMEFI58017X0009].

REFERENCES

[1] A. Rumyantsev and S. Chakravarthy, "Split-Merge Model of Workunit Replication in Distributed Computing," in Proceedings of the Third International Conference BOINC:FAST 2017., 2017, vol. 1973, pp. 27–34.

[2] Chakravarthy Srinivas R. and Rumyantsev Alexander, "Efficient Redundancy Techniques in Cloud and Desktop Grid Systems using MAP/G/c-type Queues," eng, vol. 8, no. 1, p. 17, 2018.

[3] Rumyantsev A., Chakravarthy S., Morozov E., Remnev S. (2018) Cost and Effect of Replication and Quorum in Desktop Grid Computing. In: Dudin A., Nazarov A., Moiseev A. (eds) Information Technologies and Mathematical Modelling. Queueing Theory and Applications. ITMM 2018, WRQ 2018. Communications in Computer and Information Science, vol 912. Springer, Cham.

[4] Cagdas Dirik and Bruce Jacob. 2009. The performance of PC solid-state disks (SSDs) as a function of bandwidth, concurrency, device architecture, and system organization. In Proceedings of the 36th annual international symposium on Computer architecture (ISCA '09). ACM, New York, NY, USA, 279-289. DOI: https://doi.org/10.1145/1555754.1555790

[5] Nima Elyasi, Mohammad Arjomand, Anand Sivasubramaniam, Mahmut T. Kandemir, Chita R. Das, and Myoungsoo Jung. 2017. Exploiting Intra-Request Slack to Improve SSD Performance. SIGPLAN Not. 52, 4 (April 2017), 375-388. DOI: https://doi.org/10.1145/3093336.3037728

[6] Feng Chen, Tian Luo, and Xiaodong Zhang. 2011. CAFTL: a content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives. In Proceedings of the 9th USENIX conference on File and stroage technologies (FAST'11). USENIX Association, Berkeley, CA, USA, 6-6.

[7] Qureshi, Moinuddin K., Michele M. Franceschini, and Luis A. Lastras-Montano. "Improving Read Performance of Phase Change Memories via Write Cancellation and Write Pausing". HPCA – 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture, 1–11. Bangalore: IEEE, 2010. https://doi.org/10.1109/HPCA.2010.5416645.

[8] John Ousterhout, Parag Agrawal, David Erickson, Christos Kozyrakis, Jacob Leverich, David Mazières, Subhasish Mitra, Aravind Narayanan, Guru Parulkar, Mendel Rosenblum, Stephen M. Rumble, Eric Stratmann, and Ryan Stutsman. 2010. The case for RAMClouds: scalable high-performance storage entirely in DRAM. SIGOPS Oper. Syst. Rev. 43, 4 (January 2010), 92-105. DOI: https://doi.org/10.1145/1713254.1713276

[9] Feng Chen, David A. Koufaty, and Xiaodong Zhang. 2011. Hystor: making the best use of solid state drives in high performance storage systems. In Proceedings of the international conference on Supercomputing (ICS '11). ACM, New York, NY, USA, 22-32. DOI=http://dx.doi.org/10.1145/1995896.1995902

[10] Zuo Pengfei, Hua Yu, Zhao Ming, Zhou Wen, Guo Yuncheng. (2018). Improving the Performance and Endurance of Encrypted Non-Volatile Main Memory through Deduplicating Writes. 442-454. 10.1109/MICRO.2018.00043.

[11] Ilya A. Chernov, Evgeny Ivashko, Dmitry Kositsyn, Vadim Ponomarev, Alexander Rumyantsev, Anton Shabaev. Flash-Based Storage Deduplication Techniques: A Survey// International Journal of Embedded and Real-Time Communication Systems (IJERTCS), Volume 10, Issue 3.

[12] Wu Guanying, Huang Ping, He Xubin. (2014). Reducing SSD access latency via NAND flash program and erase suspension. Journal of Systems Architecture. 60. 10.1016/j.sysarc.2013.12.002.

[13] N. Balakrishnan, "Permanents, order statistics, outliers, and robustness.," Revista Matemática Complutense, vol. 20, no. 1, pp. 7–107, 2007.

[14] N. Balakrishnan and P. C. Joshi, "A note on order statistics from Weibull distribution," Scandinavian Actuarial Journal, vol. 1981, no. 2, pp. 121–122, Apr. 1981.

[15] G. Joshi, E. Soljanin, and G. Wornell, "Efficient Redundancy Techniques for Latency Reduction in Cloud Systems," ACM Transactions on Modeling and Performance Evaluation of Computing Systems, vol. 2, no. 2, pp. 1–30, Apr. 2017.

[16] G. R. M. Borzadaran and H. A. M. Borzadaran, "Log-concavity property for some well-known distributions," Surveys in Mathematics and its Applications, vol. 6 (2011), 203 – 219.

[17] M. Bagnoli and T. Bergstrom, "Log-concave probability and its applications," Economic Theory, vol. 26, no. 2, pp. 445–469, Aug. 2005.

[18] D. G. Feitelson, Workload modeling for computer systems performance evaluation. Cambridge University Press, 2015.

[19] M. Harchol-Balter, "The Effect of Heavy-Tailed Job Size Distributions on Computer System Design," in Proc. of ASA-IMS Conf. on Applications of Heavy Tailed Distributions in Economics, Engineering and Statistics, 1999.

[20] Z. Shao and U. Madhow, "Scheduling heavy-tailed data traffic over the wireless internet," in Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th, 2002, vol. 2, pp. 1158–1162.