2016 IEEE 22nd International Conference on Parallel and Distributed Systems

# Application-Aware and Software-Defined SSD Scheme for Tencent Large-Scale Storage System

Jianquan Zhang, Dan Feng
*Wuhan National Laboratory for Optoelectronics*
*Huazhong University of Science and Technology*
*Wuhan,China*
{*zhangjianquan,dfeng*}*@hust.edu.cn*

Jianlin Gao
*Tencent Technology Shenzhen Company*
*Shenzhen, China*
*austingao@tencent.com*

Wei Tong‡, Jingning Liu, Yu Hua, Yang Gao, Caihua Fang, Wen Xia
*Wuhan National Laboratory for Optoelectronics*
*Huazhong University of Science and Technology*
*Wuhan,China*
{*Tongwei, Jnliu, csyhua, yanggao, caihuafang, xia*}*@hust.edu.cn*

Feiling Fu, Yaqing Li
*Tencent Technology Shenzhen Company*
*Shenzhen, China*
{*fufl, lyqstudy*}*@qq.com*

***Abstract*—Tencent, one of the biggest Internet companies in China, contains billions of users and over 600-PB data, and leverages thousands of SSDs in the storage system to improve system performance and obtain energy savings. Existing commercial SSDs however fail to meet the needs of the ultra largescale applications due to not matching the service patterns. In order to address this problem and deliver high performance, we propose an application-aware and software-defined SSD scheme for Tencent applications, called TSSD. TSSD explores and exploits the business characteristics of Tencent, which facilitates the efficient use of SSDs. TSSD is software-defined by packaging each flash chip as a fully independent and concurrent storage unit. Each concurrent unit can be mounted as a character device, which allows the application layer to manage the flash chips in a more efficient manner, while optimizing the data layout. TSSD further employs a host-target FTL (TFTL) that uses a dedicated interface in the application layer, which efficiently connects the application layer with flash chips. Application layer hence becomes more accurately by using the flash memory chip-level information from TFTL, including the storage utilization, the degree of wear, etc. Moreover, TFTL is a programmable FTL and provides a programmable interface to the application layer. According to the running states of SSDs and workload information, TSSD makes use of the programmable interface to efficiently improve the performance of the FTL, wear leveling, and garbage collection for the specified applications. Extensive experiments use the real-world datasets from the commercial storage systems of Tencent. The results demonstrate that TSSD significantly improves the storage system performance and meets the needs of the Tencents large-scale business applications.**

***Keywords*-Solid-State Drive (SSD); Application-Aware; Software-Defined; FTL**

## I. Introduction

In 2015, We Are Social[1] reports that the current total global Internet users has exceeded 3 billion. Tencent, as one of the largest Internet companies in China, the monthly active users of its instant messaging tool(QQ) and interactive website(QQZone) have reached 829 and 629 million, respectively. In the instant messaging mobile application of Tencent, WeChat, the monthly active users have reached 438 million. In addition, the simultaneous online users of QQ have reached more than 200 million. As a result, Tencent generates vast amounts of data every day, and the average number of accessed messages per day has reached one trillion, with about 1PB data increasing every day. Current data storage capacity has more than 600PB, while more than 50PB data are in flash memory. With so many and growing number of users, as well as the extremely large amount of data, how to ensure the IO performance for so many online requests at the same time is the key issue for Tencent storage system.

Flash is one of the novel storage devices, because of its high performance, small size, low power consumption, high concurrency, etc., flash has been widely used in the data center, enterprise storage systems, Internet services, and consumer electronics. However, in Tencents data center, we tested the general SSDs (such as, Intel SSD) for the practical applications, and found that the SSD failed to make full use of the potential of its bandwidth and storage capacity. This is because, in the real applications of Tencent, the read bandwidth utilization is about 70% to 80%, while the write bandwidth utilization is only 20% or lower. Specifically, the main reasons are threefold: First, the existing general SSDs use the centralized management of the hardware architecture and thus restrict the exertion of its parallelism, flash memory controller performance has become the bottleneck of performance. Second, the FTL in the existing general SSDs is implemented in SSDs internal firmware with the universal algorithm and thus it suffers from the problems of write amplification and unpredictable read response time.

‡ Corresponding author.

1521-9097/16 $31.00 © 2016 IEEE
DOI 10.1109/ICPADS.2016.69

Third, because operations such as garbage collection and wear leveling will trigger SSD background erase and data migration operations, resulting in negative impact on the performance of the SSDs, especially for the response time which fails to meet the requirements of Internet social network applications.

In Tencents data center, SSDs are mainly used for providing high read performance, and particularly for the important business data. When using SSDs, in order to ensure the QoS and keep all the read response time, i.e. the time spent in storage engines less than 50ms, we need to limit the write bandwidth of the SSDs, and reduce the influences to the read response time from data writing and the SSD background operations. According to the use of Intel TS8-2 SSD in Tencents data center, we find that we must limit the write bandwidth to 10MB/s after a period of using, otherwise there will be a lot of system Read Response Spikes appeared (we call the read response time which is over 50ms as a Read Response Spike. In Tencent's data center, to ensure that users access response time is within 2s, and eliminate the time spent on Internet communication, we require the device response time within 50ms).

In addition, the existing commercial general SSDs mainly aim at the general design, and academic discussion are mostly devoted to the optimization of SSD small random write performance. And in application scenario for complex business and enterprise in the market, IO model complexity, higher performance and reliability requirements, cannot give full play to the performance and service life of the flash memory particles. Tencent, as one of the largest internet companies in China with largest number of users, has diverse applications. And even in the same application, there are different types of data and obvious differences between the IO characteristics at different times, QoS requirements may also be different. According to these changes, the existing general SSDs are often unable to understand these changes, neither can make corresponding optimization strategy adjustment. Focusing on Tencents real business needs, specifically characteristics of business data access, we designed a new business sense for large-scale distributed Internet application-aware, software defined, and programmable SSD, which we call TSSD. The objective of TSSD is to reduce costs while maintaining the features including high bandwidth, low latency, service awareness, and programmable. The main contributions of this paper are as follows:

- We propose a new SSD architecture, called TSSD, it supports software defined and high concurrency. It is customized for Tencents application requirements. Specifically, each flash chip is virtualized as a character device, each of them is worked for applications independently and concurrently.
- Designed an application aware and programmable FTL mechanism, called TFTL. TFTL is a host target FTL, and it is application aware with a special interface to the application layer. The application layer sends the IO model of application workload, data features, and other semantic information to the TFTL, and it makes TFTL understand applications well.
- TFTL is a programmable FTL by providing a programmable interface for the application layer. According to the state of TFTL and workload information, TSSD load the suitable FTL policy for the applications through programmable interface, including wear leveling, bad block management, dynamic and static mapping strategy, etc.
- Under Tencent applications (such as QQ, WeChat, QQzone, etc.), TSSD keeps all the response time of the access less than 50ms, while flash memory utilization can be up to 95%, and the other 5% is for the bad block management.

The rest of the paper is organized as follows. Section 2 presents the necessary background for this research. Section 3 describes the design and implementation of TSSD. Section 4 presents and discusses our experimental evaluation of TSSD. Section 5 draws conclusions.

## II. Background and related work

### A. The Business Model of Tencent

As the largest Internet social network services company of China, Tencent has the largest amount of users, producing vast amounts of data every day. Tencent has a variety of popular applications, including the popular instant messaging tool QQ, intelligent terminal WeChat, emails, games, QQzone, WeChat moments, etc.

Currently, Tencents data volume has exceeded 600 PB, while just for the business of WeChat, the application of Friends-Circle (i.e., the moments sharing pictures with friends) has more than 40 PB data volume. In addition, the data volume of QQ photo album has reached 200 PB, personal journal of QQzone has also more than 200 TB. We analyze the log file of each business requirements to obtain the characteristics of data I/O requests to Tencents applications and thus divide the Tencents applications into three categories: (1) read-intensive applications that the read requests is the majority, such as the personal portrait, users pictures, etc. (2) read & write intensive applications, such as QQ chat and WeChat. (3) write-intensive application, such as QQ image index file. We select three kinds of applications and record the I/O requests distribution as shown in Figure1.

As shown in Figure 1, The first application is the index of QQ album, and the average size of the read request and write request are about 1KB and 5KB respectively, while the average ratio of the read and write requests is about 12:1. Therefore, this is a typical read-intensive application. The second application is WeChat, and the average request size of read and write are about 32KB and 22KB respectively

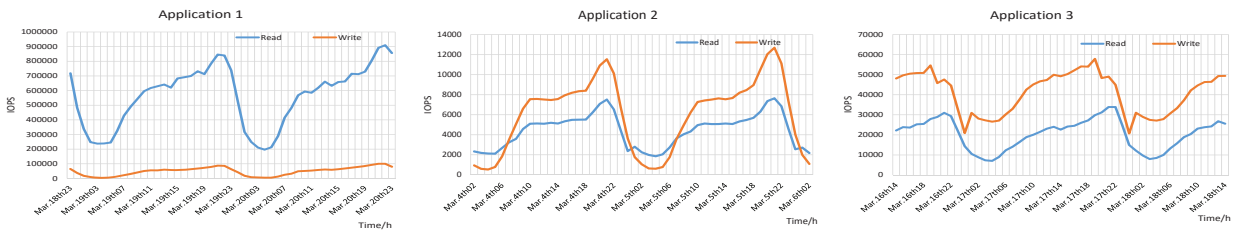


Figure 1. The IO requests distribution of the three typical business applications in Tencent.

while the proportion of read and write requests is about 2:3. And the third application is the index of Friends-Circle on WeChat, the average read and write request size are 250B and 147B respectively while the proportion of read and write requests is about 1:2, this is a typical small write-intensive application.

For the typical social networking applications of WeChat, QQ, and QQzone in Tencent, the user experience and QoS are the most important consideration in their storage systems, and the system response time, especially the read response time will directly affect the user experience. Therefore, Tencent storage systems employs SSDs to improve the access performance and quality of service. However, exploring hardware potential of SSDs to satisfy the business requirements of Tencent is still a challenge as detailed in the next subsection.

### *B. Universal SSD Architecture & Deficiencies*

Currently, the general SSD in the market is mainly of two types, one is with SATA or SAS interfaces for individual PC and some small-scale servers, the another uses the PCIE interface which is faster than SATA and mainly used in the servers[2]. The architecture of general SSD is shown in Figure 2.

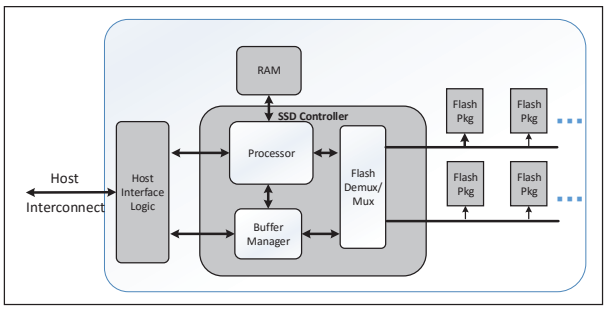


Figure 2. The architecture of general SSD.

Flash SSD mostly consists of three parts: Flash memory chip, Interface (connecting the flash memory chip and the host side), Flash memory Controller. Flash Translation Layer (FTL), as the most important part of SSD, is mainly implemented in the flash memory controller. Flash memory chip has three basic operations, read, write, and erase. Wherein reading and writing the flash memory pages are basic operation unit, and the erase operation must be in units of blocks. Flash SSD has the erase-before-write problem, which stems from the intrinsic physical nature of flash memory. Erase-before-write requires that an occupied data block (typically 64 or 128 of 4KB memory cells called pages) must be erased before the new data can be written to that block [3]. And host interface section mainly includes SATA, SAS and PCIE interface. Initially, SSD uses the SATA interface to be compatibility with existing file systems, applications. With multi-channel flash memory and other technologies for the development of mining properties, so the interface is becoming an important factor restricting the flash performance, and therefore began to use the faster PCI-E interface in high-performance applications. For general SSD, its flash memory controller is implemented mainly in the embedded processor or FPGA of which the computing power is limited. However, FTL implemented in flash memory controller, the complex computing tasks (including address mapping, garbage collection and wear leveling, etc.), such that the processor computing power has become a constraint SSD performance factors. Based on this, Fusion IO first proposed based on the host side FTL which will move to the host side implementation, in order to make greater use of host computing resources and memory resources[3].

Most applications use flash SSD as a block device compatibility with HDD, or faster HDD at the system level[4]. It did not give full play to the characteristics of flash memory, but increased the complexity of the system software and hardware. In addition, SSD is a black box for the application layer, its internal organizational structure and corresponding software optimization strategies, and often is considered a trade secret protection, making it more difficult for the application layer optimization flash characteristics[5]. Because of SSD internal operations including garbage collection, wear leveling and other background operation, its real-time performance becomes unpredictable, especially the request response time[6].

### *C. Related Work*

Due to the variable garbage collection latency, NAND flash memory storage systems may suffer long system response time, especially when the flash memory is close to be full. Most of existing flash translation layer (FTL) schemes focus on improving the average response time but ignore to provide a desirable response time even at worst

case. In Tencents storage systems, when the read response time is up to 50ms, we count it as Response Spike. Shao et al.[7] proposed a Real-time Flash Translation Layer(RFTL) scheme to hide the long garbage collection latency. Chang[8] and his group proposed real-time garbage collection for flash memory storage systems, where predictable performance is guaranteed by ensuring enough free space is always available for write requests.

They improved the average response time, but they still cant avoid the negative impact of read response time by write and erase operations, which will cause response spike.

SSD performance and lifetime are highly workload-sensitive[5], and in general SSD cant work well for all kind of workloads. Shin[9] and his group designed a Reconfigurable High Performance SSD(RHPSSD) prototype, and they found that the performance metrics will vary from several percent to more than tens of times among each other depending on the decision made for designing each FTL functionality.

Current FTL schemes have inevitable limitations in terms of memory requirement, performance, garbage collection over-head, and scalability. Wei[10] and his group proposed a workload adaptive flash translation layer as WAFTL. WAFTL explores either page-level or block-level address mapping for normal data block based on access patterns, random or sequential. Du etc.[11] proposed a Convertible Flash Translation Layer (CFTL) which is a hybrid flash translation layer adaptive to workloads so that it can dynamically switch its mapping scheme to either a page level mapping or a block level mapping scheme. Nitro[12] uses a block size that is aligned with flash erasure block size. However due to different data has different attributes, with different update frequency, using block mapping without good data layout not only cannot reduce the fragmentation, but will increase the block erase and frequent data migration operation.

For programmable SSD, Steven Swanson etc. present a user-programmable SSD-Willow, which provides SSD APP programming interface[13]. This helps users to build their own APP or call basis APP through a programming interface to achieve flexible use SSD. However, Willow needs to use multiple SPU (storage processor unit) embedded processors manage storage media in the hardware system architecture, and run the SPU-OS operating system on this basis, which greatly increased the complexity of the system and made it more difficult to achieve the system maintenance and scalability. For the new SSD architecture, Jian Ouyang et al.[14] propose software-defined flash (SDF), a hardware/software co-designed storage system to maximally exploit the performance characteristics of flash memory in the context of Baidu workloads that mainly including online/offline key-value storage. In order to enhance storage performance of the SDF, it shortens the SSDs storage IO path by simplifying the IO protocol stack mode. In addition, Each SDF consists of a customized flash device wrapped with a layer of software providing a symmetric block access interface. So, its essence is similar to a number of small capacity SSD encapsulation integration and it has not completely addressed the disadvantages of general SSD. Since the SDF is specifically designed for the application of Baidu, and there is no corresponding adjustment depending on the application of optimization strategies, and therefore cannot be directly applied to the storage system of Tencent.

## III. System design

Design goals: according to the demand characteristics of our application, we need to design a specific SSD (TSSD) which provides high-performance, high-bandwidth, low-latency for the applications of Tencent to meet the business requirements of Tencent and TSSD is aware of the variety of Tencents business and adjusts strategies dynamically. Since the property of Tencent social networking Internet applications requiring low latency and predictable response time, especially for read response time, as it will directly affect user experience. Meanwhile, since flash memory is still relatively expensive, TSSD needs to provide higher utilization, including utilization of storage space and bandwidth. From the perspective of system development, TSSD needs to have better scalability.

Based on above design goals, we designed a highly-parallel architecture abstraction which can make full advantage of flash hardware parallelism in order to meet demand for the storage service of Tencent, which requires high performance and low latency. Considering application characteristics and I/O properties of Tencent, we have designed a host-based FTL (TFTL) which is based on the above architecture. By using the interactive interface between TFTL and application layer or flash memory, the application layer of TSSD is aware of flash memory and TFTL is aware of the applications. Therefore, it can maximize the parallelism of SSD to improve system performance and reduce latency. In order to achieve controlled access latency, TFTL calculates load of each unit, utilization of storage space and bandwidth for the system application layer. So that the application layer is aware of flash memory chips and makes reasonable schedule. Furthermore, in order to reduce the impact of request response times caused by background operations of flash (including garbage collection and wear leveling), the application layer chooses appropriate time to initialize garbage collection and erase operations depending on how busy the application is and the utilization of flash space. Wear leveling is also triggered based on configuration requirements by the application layer, which chooses corresponding wear leveling strategies at the system level.

### A. The Architecture of TSSD

a) Design and development of a highly parallel architecture. In order to meet the needs of high bandwidth, low

latency and high concurrency of Tencent internet application service, and to improve performance and storage IOPS and degree of parallelism, we designed a fully parallel architecture from an architectural perspective, as shown in Figure 3. TSSD is composed of three FPGAs and some flash, which connects to the host by PCI-E 2.0 X8 interface. FPGA is connected to each other via 10Gb SerDes interface. Each FPGA controls 17 flash channels with a separate DRAM cache, and it allows each channel to be abstracted into four character devices by using the drive and CSL layer. So it enables each board presenting 204 memory cells which are concurrent and under unified control by TFTL to the application layer, as shown in Figure 3. We implement a simple interface and the command distribution in the FPGA, TFTL is achieved in the host.

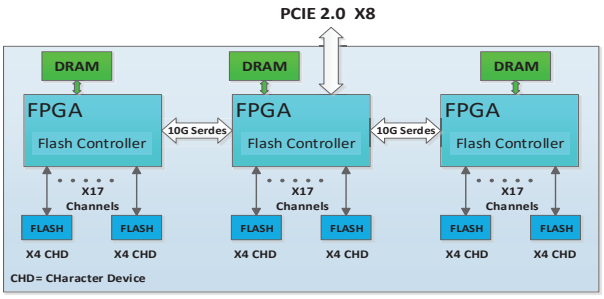


Figure 3. Hardware architecture diagram of TSSD.

In order to meet the needs of high bandwidth, low latency and high concurrency of Tencent internet application service, and to improve performance and storage IOPS and degree of parallelism, we designed a fully parallel architecture from an architectural perspective, as shown in Figure 3. TSSD is composed of three FPGAs and some flash, which connects to the host by PCI-E 2.0 X8 interface. FPGA is connected to each other via 10Gb SerDes interface. Each FPGA controls 17 flash channels with a separate DRAM cache, and it allows each channel to be abstracted into four character devices by using the drive and CSL layer. So it enables each board presenting 204 memory cells which are concurrent and under unified control by TFTL to the application layer, as shown in Figure 3. We implement a simple interface and the command distribution in the FPGA, TFTL is achieved in the host.

b) Software defined TSSD. We add CSL(Common SSD Layer) between TFTL and hardware layer and packaged SSD devices into tfd_mtd device which behaves as a character device to the application layer. The TFTL layer is provided with a unified driver interface so that it can fits different customized hardware. Hardware offered by different hardware vendors can be applied to the system so long as they meet the requirement of uniform CSL standardized interfaces. Therefore the system has good scalability and compatibility, and also reduce reliance on hardware vendors. TSSD packages SSD into multiple MTD devices by using

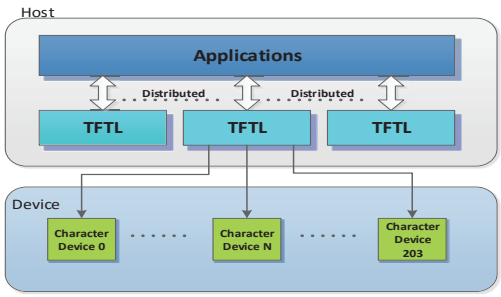


Figure 4. The Logic structure diagram of TSSD.

the MTD framework in Linux, which shows the application layer as a character device. Each MTD device provides character device operation interface (e.g. open, read, write, close, etc.) and a unique user interface (e.g. erase) for flash memory taking characteristics of flash memory into consideration. Since MTD interface is a standardized interface, TFTL can support almost all state-of-art flash memory chips in the market. TSSD provides several user interfaces to upper software as a character device, several character devices to achieve wear leveling and IO scheduling in the address space of entire SSD card.

### *B. Application-aware TFTL*

a) Host-based FTL. First of all, TFTL is based on the host, its starting point is fundamentally different from state-of-art host FTL. Existing host FTL mainly make use of strong computing power and abundant memory resources of the host, FTL is moved to the host to address inadequate computing capability shortcomings of embedded processors. In our design, in addition to facilitate the use of host-side computing and memory resources, there is a very important consideration, i.e. it is easy for TFTL to make information exchange with application layer, which will be discussed later (TFTLs business awareness of application and application awareness of flash memory chips. TFTL is implement in the host, so that the communication between TFTL and application layer becomes more convenient, furthermore effectively avoid negative impacts of interactive information transmission between application layer and TFTL on user access data.

b) Flash-aware to application layer. From the perspective of the overall system, in TSSD architecture, TFTL has a real-time statistics such as how busy each concurrent device is and space utilization, degree of wear and other information related to physical flash memory device, and TFTL reports to the application layer in time, so that the application layer to be better aware of physical device. In distributed systems, combining with consideration of the properties of flash memory, TSSD makes appropriate data layout scheduling, IO balancing and other relevant strategies to adapt better to business features that enable controlled delay of user service requested, and effectively reduce data access latencies and

the occurrence of response spikes.

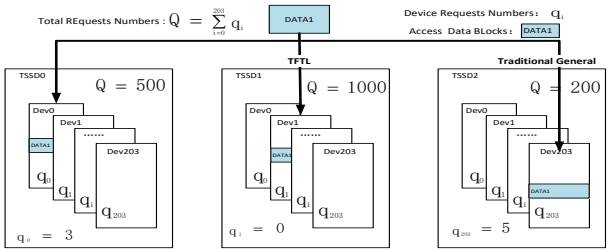


Figure 5. Application layer perception of Flash chip.

As shown in Figure 6, when the application layer requests to read data DATA1, after looking up it founds DATA1 respectively in three copies in the distributed system exists, i.e. TSSD0 / Dev0, TSSD1 / Dev1, TSSD2 / Dev203. TFTL then respectively inquiry load request queue of TSSD0, TSSD1, TSSD2 and total length returned is respectively $Q_0$=500, $Q_1$=1000 and $Q_2$=200, while load request queue length of TSSD0 / Dev0, TSSD1 / Dev1, TSSD2 / Dev203 are respectively $q_0$= 3, $q_1$= 0, $q_{203}$= 5. For general SSD, usually based on $Q_2$, $Q_0$, $Q_1$, thus the request is assigned to TSSD2, when a request arrives TSSD2 and the system find out TSSD2 / Dev203 is busy, the request is mounted under corresponding request queue in Dev203 to wait. For TFTL, the application layer gets the conclusion of $q_1$,$q_0$, $q_{203}$ based on load information of concurrent device request queue provided by TFTL, thus directly assigned request to TSSD1, whereas TSSD1 / Dev1 is idle this time. It can respond data requests of DATA1 immediately, thus effectively reduces the direct request response time of DATA1.

In addition, the application layer can also be aware of space utilization of flash memory storage in TSSD, degree of wear of corresponding SSD nodes and other related information through TFTL, and thus play an important role in the data layout of application layer distributed system, wear leveling and other strategies.

c) Applications-aware TFTL (Business-Aware). Different types of business have different service attributes, thus there is a considerable difference between their IO characteristics. In addition, for the same service in different time periods, its IO characteristics, access frequency, load pressure and so on are also different.

TFTL is aware of applications, i.e. the host TFTL can receive data semantic information of the application layer through the interface of the application layer. Then it can get service attributes and IO characteristics of data access faster and more accurate, including the degree of cold of requesting data, read-primarily types, often-updated types and other related properties, thereby enabling the corresponding data layout and scheduling optimization based on characteristics of flash memory. We classify the services on Tencents SSD and extract IO features, TFTL can depend on the access IO model of the application and load intensity, with TFTL programmable interface to respective set different FTL policy, so as to better meet the requirements of application service, improve application service capabilities. In SSD, identification of hot and cold data is also very important. It is so crucial for data layout, IO scheduling, caching strategies, load balancing, and wear leveling in flash. The application layer can identify the extent of the hot or cold data more quickly and accurately based on relevant semantic information of the application. d) Strictly-controlled read request response time. Since the erase operation in the flash memory systems cost more time, usually two orders of magnitude than write operation, erase operation has a great negative impact on response time of flash service. In order to ensure user experience, we strictly control the request response time of user data, remove affect to user requests response time caused by erase and other background operations. The application layer chooses appropriate time to send erase command to TFTL layer actively according to busy degree of its own services, busy degree of physical device is each storage node and degree of I/O load balancing. And when system is idle, it initiates wear leveling, static address mapping, bad block management and other related strategies depending on the characteristics of applications. Furthermore, since erase operation is initiated by the application layer, and flash device is abstracted through concurrency, all physical space of TSSD can be identified and used by the application layer without additional redundancy space, thereby increase the utilization of flash memory board, and reduce costs.

### *C. Programmable interfaces for TSSD*

In order to meet the requirements of different business services and changes of business characteristics which may happen over time, we specifically designed programmable interface for the application layer in TFTL. The application layer can customize corresponding FTL strategies including wear leveling, bad block management, static mapping, dynamic mapping, and other related strategies according to the characteristics and needs of business through programmable interface. For the traditional commercial SSDs FTL, its internal management policy for is invisible to user and application layer and cant be changed after the device leaves the factory. Commercial SSD typically modeled flash as a virtual block device, which has good universality. But it also has obvious shortcomings, i.e. mediocre performance, poor flexibility, etc. Ordinary commercial SSD generally need to consider the universality of its products, so optimization for particularly specific business is not considered. The optimization in the consumer market mainly concentrates on performance of small random IO read. While in the complex business scenarios and enterprise markets, IO model is complex and needs higher performance and reliability, traditional model cannot make best of the characteristics of performance and life of NAND-flash. Existing Host-based FTL implementation has similar function to target-based

FTL, needs to do specific optimization for a specific business model with a long cycle of development import, it cannot switch dynamically according to the business load. Our TFTL provides a special configuration select interface for the application layer, application can choose most appropriate algorithm according to their own memory access characteristics, or load custom algorithm, targeted to enhance the performance and reliability. Moreover, applications can query the state of TFTL by interface, and according to their own business load and running information of TFTL, invoke the TFTL management structure to make adjustments to achieve dynamic adjustment of the performance and reliability.

## IV. Performance Evaluation

In this section we evaluate the performance of TSSD in real Application storage systems of Tencent. Currently there are more than 30PB of TSSDs deployed in the storage systems. TSSDs have been used for many applications more than two years and a large amount of performance data has been collected from the monitoring and profiling system. Firstly, We will first describe the experimental setup, and then we give the evaluation results under simulation environment and Tencent production workloads.

### A. Experimental Setup

The TSSD which we currently used on the Real storage systems has a capacity of 1.5TB, and almost all of them are utilized by applications, except small part of them are converted into bad block management. TSSD adopts chip capacity of 8GB, and whose page size is 4KB and block size is 512KB. TSSD connect to the host with the interface of PCI-E 2.0x8. The test host have two 2.00GHz Intel Xeon E5-2420 CPU, and 32GB main memory. And each node mounts two TSSDs. In the evaluation we use the Fusion-IO IO-Driver[15] and Intel TS8-2 for comparison. The capacity of Intel TS8-2 is 300GB and 11 TS8-2s connect to a single host with SATA interface. while the capacity of Fusion-IO is 640GB and with an interface PCI-E 2.0x4.

With the microbenchmark test, we used a test tool Presscall. Presscall that designed by Tencent is aimed at satisfying the application requirement of Tencent. It can be installed in the equipment running on different number of read threads or write threads. Through limiting the write bandwidth, we can get the read/write IOPS, average response time, and the ratio of Read response spike under the condition of different write bandwidth. For simulating the real application environment, we write sequential requests of 512 KB to the whole SSD, then run the random write requests of 4 KB to fragments of the SSD for one hour.

For the Real workload test, we used an open source tool of TCPCOPY, which is an online request replication tool, also a TCP stream replay tool. It fits for real testing, performance testing, stability testing, stress testing, load testing, and smoke testing, etc. We use it to derive the request data flows from the real system, and concurrently send them to different hosts with TSSDs or other devices. We hence examine the performance of TSSD and general SSD under the same configuration. In order to achieve test result, we do multiplication of the data flows to test TSSD and general SSD with larger load.

### B. Experiments with Microbenchmarks

In order to verify that TSSD has a good ability to adapt different IO model, we use the tool of Presscall, which is good at simulating the behavior of Tencents application workload. Firstly, we build three storage system, which separately used a single TSSD, two Fusion IO SSD and 11 Intel TS8-2 SSDs. We collect their read IOPS, the Average Read Response time and Read Response Spikes(read response time over 50ms) under the limitation of different write bandwidth. The results are shown in Figures 6 and 7.

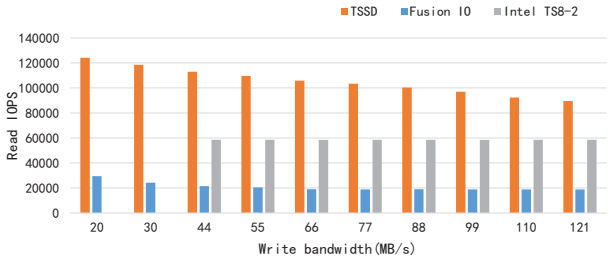


Figure 6. Read IOPS of multiple SSDs, under different write bandwidth.(The IOPS of Fusion IO and TSSD are tested on one board, the data of Intel TS8-2 is tested on a cluster of 11 Intel TS8-2 SSDs ).

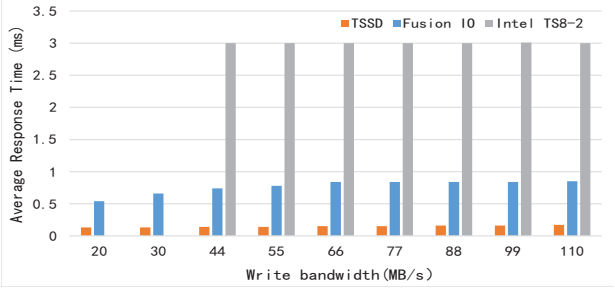


Figure 7. Average response time of multiple SSDs, under different write bandwidth.

As the software defined TSSD, every flash chip in TSSD can be accessed currently, and it significantly improves the parallelism and read performance of TSSD. In addition, due to the TFTL with Flash awareness and application awareness, it obviously reduced the negative effects to the Read response time from write request and background operation such as erase and Wear-leveling. It reduced the average read response time, and almost eliminates the read response spike. For the general SSD, Fusion IO and Intel TS8-2, because of the existence of the background operation, which makes systems have many read response spikes, and

with the write bandwidth increasing, the trigger background erase operation will be more frequent, so as to make the read response spike has increased dramatically. As shown in Figure6, under different write bandwidth, TSSD has the highest read IOPS, on average it is improved 1.5 times relative to Fusion IO, 70% improved compared to Intel TS8-2. For average read response time, TSSD is also the best one. The average read response time of TSSD is stable, and it is maintained at 0.13ms to 0.17ms, the maximum read response times are within 10ms, 99.9% of read response times are controlled within 3ms, without read response spike. While for the Intel TS8-2, a lot of read response spike are generated when the write bandwidth is only 44MB/s, and because of the limitation of hardware, the average response time of Intel TS8-2 SSD are stable in 3ms.

### C. Test with Real Workloads in Tencent

In this section, we use the tool of TCPCOPY, to export the real data flow from the reality online network, and meantime we respectively deliver to the hosts equipped with guide to TSSD and general SSD (Intel TS8-2), then we directly compare their performance, recording the average respond time and their response spikes under the same conditions. For achieving the results in our evaluation, we respectively do the multiplication operation of data sets to the data stream so as to compare the performance between general SSD and TSSD under the higher load conditions. In addition, since the performance of the SSD exists big difference between the blank disk and the filled with data, for achieving the desired real test results, we advanced made SSD fragmentation and data embedded operations before testing the data flows (i.e., firstly fill the SSD with random data, then advanced write the data acquired by read request to the SSD).

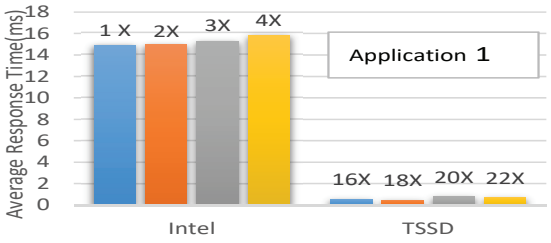


Figure 8. The Average Read Response time of application 1 under multiple times of load(1X denote as 1 time of load).

For a variety of applications in Tencent, we choose three typical applications (the QQ album index, the chats recording of WeChat, and index of WeChat friend circle) for the real diversion test, and we have the detailed analysis for the characteristics of these three applications in Chapter2.

Application 1(the QQ album index) is a typical read-mostly applications as its ratio of read and write requests is about 12:1. We use TCPCOPY to make diversion from an online web server, its average IOPS is 22005. As shown in Figure8, the average response time of Intel SSD is close to 15ms while we treat 1 time of its load as the baseline. However due to the highly concurrent software-defined architecture of TSSD and high performance brought by its superior hardware, the average responding time of TSSD with load of 45 times(it has reached the maximum limit of network bandwidth) to the baseline is much lower than that of Intel SSD with load of 1 time to the baseline.

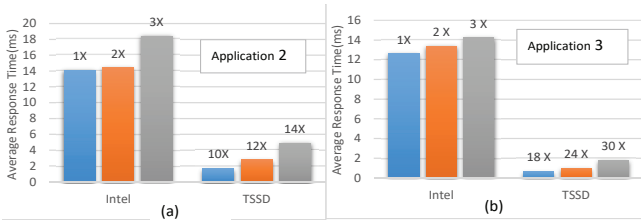


Figure 9. The Read Response Spike Rate of application 2&3 in Intel TS8-2.

WeChat chat (C2C) is a typical application with frequent reading and writing operations, which is also one of the most important applications of Tencent. Its average IOPS is 26712 with load of 1 times to the baseline. As shown in Figure9(a), the average responding time of Intel SSD is close to 14ms at 1 time of load, while that of TSSD with load of 14 times to the baseline is only 5ms. As Figure9(b) shows, application 3 (index of WeChat friend circle), the average responding time of Intel SSD is close to 13ms while that of TSSD with the load of 30 times to the baseline is only 2ms.

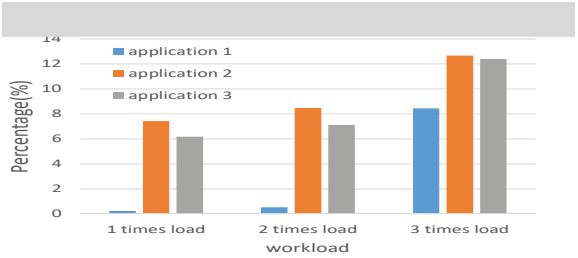


Figure 10. The Read Response Spike rates of application 1,2&3 in Intel TS8-2 under different times of load.

As shown in Figure10, for the application 1, although the Read Response Spike rate of Intel SSD with load of 1 time or 2 times of the baseline is less than 1%, but it is surged more than 8% when the load is 3 times to the baseline, 9% when the load is 4 times to the baseline and it cannot satisfy Tencent high requirement to the read responding requests any more. When we uesed the Intel SSD for the Application 2&3 the Read Response Spike rate are more than 6%. While the Response Spike Rate of TSSD with load over 22 times to the baseline is less than 0.01%, it can be basically negligible for all of the applications.

Through the evaluation used TCPCOPY to do the test with the multiple load to baseline, due to the high concurrency brought by the software-defined storage architecture of TSSD, and the high performance brought by its

Application-aware and Flash-aware characteristics, it makes the application layer have convenient memory management and data distribution for each single chip, and it reduces the influence brought by the operations such as garbage collection on responding time of reading request. Through the evaluation, the average response time of TSSD is much less than that of Intel SSD, the most important is that the ratio of Read Response Spike (response time of read request ¿ 50ms) is maintained less than 0.1%, it greatly satisfy the Tencent applications; although the average response time of general Intel SSD meets the basic needs of the Tencent business, its respond spike rate increases with the increase of load, it cannot meet the system requirements in reality (the Response Spike Rate required in Tencent applications is not more than 0.1%). In addition, for different applications, such as the QQ album index, the chats recording of WeChat, and index of WeChat friend circle, the general Intel SSD with load of 1 time to baseline under application 1 can basically meet the system requirements, but its responding glitch with load of 1 time to baseline among other two applications is far beyond the system requirement. While the TSSD is able to perceive different application characteristics, adjust the storage policy through providing a programmable interface, thus always meeting different applications.

## V. Conclusion and Future Work

In this paper, we present TSSD, a software-defined SSD for Tencent‘s large scale Internet applications. TSSD is a high performance and concurrent SSD architecture, and it also has an intelligent and programmable FTL, called TFTL. TSSD is software-defined and employs TFTL to manage the hardware efficient and intelligent. Our experimental measurements show that TSSD has the best performance compare to Fusion IO and Intel SSDs, at the metrics of IOPS, Average response time, the distribution of response time, and the latency spikes. It has been successfully and widely used in Tencent‘s storage systems, and met the needs of Tencent very well.

Although TSSD has met the requirements of Tencent services, there is still room for improvement. When TSSD serves for the write-intensive workloads, the overhead of CPU at the host target can’t be ignored. In the future, we are committed to reducing the overhead of the host CPU. In order to dig up the potential of the hardware and software of SSDs, we will analyze the workload characteristics, and then we design new FTL strategy for the TFTL and workloads more suitable.

## Acknowledgment

This work was supported by the National Basic Research 973 Program of China under Grant No. 2011CB302301; 863 Project No. 2013AA013203, No. 2015AA015301, No. 2015AA016701; NSFC No. 61173043, No.61303046, No. 61472153, No. 61502190; This work was also supported by Tencent and Key Laboratory of Information Storage System, Ministry of Education, China.

## References

[1] S. Kemp, “Digital, social & mobile in apac in 2015,” http://wearesocial.sg/blog/2015/03/digital-social-mobile-in-apac-in-2015/, 2015.

[2] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and C. Ren, “Exploring and exploiting the multilevel parallelism inside ssds for improved performance and endurance,” *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1141–1155, 2013.

[3] Fusion-IO, “Fusion-io drive datasheet,” www.fusionio.com/data-sheets/iodrive-data-sheet/.

[4] I. Petrov, G. G. Almeida, A. P. Buchmann, and U. Gräf, “Building large storage based on flash disks.” in *ADMS@ VLDB*, 2010, pp. 34–42.

[5] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. S. Manasse, and R. Panigrahy, “Design Tradeoffs for SSD Performance.” in *USENIX Annual Technical Conference*, 2008, pp. 57–70.

[6] W. Shin, M. Kim, K. Kim, and H. Y. Yeom, “Providing QoS through host controlled flash SSD garbage collection and multiple SSDs,” in *International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2015, pp. 111–117.

[7] Z. Qin, Y. Wang, D. Liu, and Z. Shao, “Real-time flash translation layer for nand flash memory storage systems,” in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th*. IEEE, 2012, pp. 35–44.

[8] L.-P. Chang, T.-W. Kuo, and S.-W. Lo, “Real-time garbage collection for flash-memory storage systems of real-time embedded systems,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 4, pp. 837–863, 2004.

[9] J.-Y. Shin, Z.-L. Xia, N.-Y. Xu, R. Gao, X.-F. Cai, S. Maeng, and F.-H. Hsu, “Ftl design exploration in reconfigurable high-performance ssd for server applications,” in *Proceedings of the 23rd international conference on Supercomputing*. ACM, 2009, pp. 338–349.

[10] Q. Wei, B. Gong, S. Pathak, B. Veeravalli, L. Zeng, and K. Okada, “Waftl: A workload adaptive flash translation layer with data partition,” in *Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on*. IEEE, 2011, pp. 1–12.

[11] D. Park, B. Debnath, and D. H. Du, “A workload-aware adaptive hybrid flash translation layer with an efficient caching strategy,” in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*. IEEE, 2011, pp. 248–255.

[12] C. Li, P. Shilane, F. Douglis, H. Shim, S. Smaldone, and G. Wallace, “Nitro: A capacity-optimized ssd cache for primary storage,” in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia, PA: USENIX Association, Jun. 2014, pp. 501–512. [Online]. Available:

[13] S. Seshadri, M. Gahagan, S. Bhaskaran, T. Bunker, A. De, Y. Jin, Y. Liu, and S. Swanson, “Willow: A user-programmable ssd,” in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 2014, pp. 67–80.

[14] J. Ouyang, S. Lin, S. Jiang, Z. Hou, Y. Wang, and Y. Wang, “SDF: software-defined flash for web-scale internet storage systems,” in *ACM SIGPLAN Notices*, vol. 49, no. 4. ACM, 2014, pp. 471–484.

[15] I. Fusion-io, “Fusion-io iodrive duo,” http://www.fusionio.com/- products/iodriveduo.