A Proactive Fault Tolerance Scheme for Large Scale Storage Systems

Xinpu Ji, Yuxiang Ma, Rui Ma, Peng Li, Jingwei Ma, Gang Wang, Xiaoguang Liu(✉), and Zhongwei Li(✉)

College of Computer and Control Engineering, Nankai University, Tianjin 300350, China
{jixinpu,mayuxiang,marui,lipeng,mjwtom,wgzwp,liuxg, lizhongwei}@nbjl.nankai.edu.cn

Abstract. Facing increasingly high failure rate of drives in data centers, reactive fault tolerance mechanisms alone can hardly guarantee high reliability. Therefore, some hard drive failure prediction models that can predict soon-to-fail drives in advance have been raised. But few researchers applied these models to distributed systems to improve the reliability.

This paper proposes SSM (Self-Scheduling Migration) which can monitor drives' health status and reasonably migrate data from the soon-to-fail drives to others in advance using the results produced by the prediction models. We adopt a self-scheduling migration algorithm into distributed systems to transfer the data from soon-to-fail drives. This algorithm can dynamically adjust the migration rates according to drives' severity level, which is generated from the realtime prediction results. Moreover, the algorithm can make full use of the resources and balance the load when selecting migration source and destination drives. On the premise of minimizing the side effects of migration to system services, the migration bandwidth is reasonably allocated. We implement a prototype based on Sheepdog distributed system. The system only sees respectively 8 % and 13 % performance drops on read and write operations caused by migration. Compared with reactive fault tolerance, SSM significantly improves system reliability and availability.

Keywords: Proactive fault tolerance · Distributed storage system · Priority scheduling · Data migration · Resource allocation

1 Introduction

With the development of information technology, the scale of the storage system is increasing explosively. Drives are the most commonly replaced hardware component in the data centers [1]. For example, 78 % of all hardware replacements were caused by hard drives in data centers of Microsoft [1]. Moreover, block and sector level failures, such as latent sector errors [2] and silent data corruption [3], cannot be avoidable when the capacity of the whole system becomes larger and larger.

G. Wang et al. (Eds.): ICA3PP 2015, Part III, LNCS 9530, pp. 337–350, 2015.
DOI: 10.1007/978-3-319-27137-8_26

Since drive failures have become a serious problem, lots of studies have been focusing on designing erasure codes or replication strategies to improve storage system reliability. They are typical reactive fault tolerance methods used to reconstruct data after failures occur. However, due to the high cost, these methods cannot meet the demands of the high service quality in data centers.

To reduce the reconstruction overhead and improve the system reliability, proactive fault tolerance was proposed, which enables the actions to be taken before failures happen. Hard drive failure prediction models are proposed firstly as a typical proactive fault tolerance strategy. These models use some statistical or machine learning methods to build prediction models based on the SMART attributes [4]. Some of them have reached a good prediction performance. For example, a model using classification Tree (CT) could predict over 95 % of failures at a FAR under 0.1 % on a real-world dataset containing 25,792 drives [5].

Although failure prediction is of great significance, the ultimate goal of prediction is to adopt some reasonable strategies to handle the failure prediction results. Recently, some studies applied failure prediction models to distributed systems, such as Fatman [6] and IDO [7], but they simply migrated dangerous data by using prediction results without taking priority and the migration impact on the system performance into consideration.

In this paper, we develop SSM (self-scheduling migration), which collects drives' status to determine their severity levels and uses a pre-warning handling algorithm to protect the in danger data. There are several issues needed to be addressed by the algorithm. How to fully use the system resources to migrate the data as soon as possible. How to dispose the drives in different severity levels differently. How to reduce the impact on the normal service. More importantly, the algorithm also balances the migration load evenly to each drive, which guarantees a stable quality of service when data is scattered evenly.

In summary, our main contributions are:

- Design *Monitor* and *Predictor* module which can monitor drives' health status in distributed systems using prediction models and then determine drives' severity levels.
- Propose an pre-warning handling algorithm to achieve high availability and reliability.
- Apply SSM to Sheepdog and evaluate the benefit.

The rest parts of this paper are organized as follows. Section 2 surveys related work of fault tolerance in storage systems. Section 3 illustrates the design of SSM. We will present the experimental results in Sect. 4. Section 5 concludes the paper.

2 Related Work

Storage is a critical component in data centers and how to ensure their reliability becomes a popular topic in the storage community. In the late 1980s, RAID technologies, such as RAID-1 and RAID-5, were firstly proposed as a fault tolerance

mechanism and have been widely used in the disk array [8]. Blaum et al. [9] proposed EVENODD which is the first double-erasure-correcting parity array code based on exclusive-OR operations. The computational complexity of this kind of codes is far lower than that of RS code.

With the development of cloud storage, hard drive failures are common which need to be handled. So the data recovery performance becomes increasingly important. In cloud storage systems, network and disk I/O, have a great influence on the service. Consequently, recent research on reliability focused on how to reduce I/O overhead incurred by data recovery. For example, Cidon et al. proposed Copyset [10], which limits the data replicas within node groups rather than over all storage nodes. This strategy reduces the data loss probability and the recovery overhead effectively. Quite a few methods [11] are also proposed to improve the recovery performance for cloud storage systems using erasure codes. Weaver code [12] and Regenerating code [13] all reach a balance between space utilization and disk/network I/O overhead.

Either replication or erasure coding are typical reactive fault tolerant techniques. Even with aforementioned methods, they hardly can provide satisfactory reliability and availability with low cost. On the contrary, proactive fault tolerance can predict drive failures in advance and therefore provide enough time for the operator to take actions before failures really occur. At present, Self-Monitoring, Analysis and Reporting Technology (SMART) is implemented inside of most hard drives [4]. The threshold-based method can only obtain a failure detection rate (FDR) of 3–10 % with a low false alarm rate on the order of 0.1 % [14]. So researchers have proposed some statistical and machine learning methods to improve the prediction performance. Especially in [5], Li et al. presented hard drive failure prediction models based on Classification and Regression Trees, which perform better in prediction performance as well as stability and interpretability.

Hard drive prediction models are intended to be used in real-world storage systems to improve reliability and availability. However, only a few researchers focused on how to use the predictions (pre-warnings) to improve the reliability of the real storage systems. IDO can find the soon-to-fail disk, migrate proactively data of hot zones to surrogate RAID set [7]. Once a disk fails, it reconstructs hot data with surrogate set and recovers cold data with RAID mechanism on the failed disk. However, IDO is not designed for distributed storage systems and locality implementation needs information from the superior file system. RAIDSHIELD [15] uses the threshold-based algorithm to predict single drive failures and prioritizes the most dangerous RAID groups according to joint probability.

In this paper, we present SSM, which is an comprehensive system, employing pre-warning handling algorithm combined with drive prediction models. Moreover, we apply the mechanism into Sheepdog to evaluate the effectiveness.

3 Architecture and Design

Figure 1 depicts the architecture of our proactive fault tolerant system, namely SSM. It consists of five functional modules: *Monitor*, *Collector*s, *Predictor*,

Trainer and *Scheduler*. *Monitor* is used to monitor the status of individual drives. *Collector*s are responsible for collecting SMART information from *Monitor*. *Predictor* assesses drives' severity levels based on failure prediction models. The function of *Trainer* is updating prediction models periodically to prevent them from aging. *Scheduler* as the kernel module in SSM, manages and schedules data migration tasks with different severity levels. It is composed of three parts: priority-based scheduling, multi-source migration and bandwidth allocation.

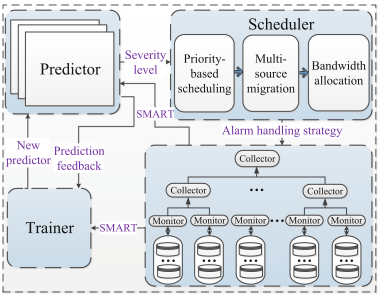

Fig. 1. The architecture of SSM.

In order to achieve portability, *Monitor*, *Collector*s, *Trainer* and *Predictor* are designed as four independent modules. They expose several interfaces (as Table 1 shows) that can be used by other modules in the system. Due to their independence, the interfaces also can be used in other distributed systems to implement their own migration algorithm.

Table 1. The interfaces exposed by *Monitor*s and *Predictor*.

Interface name	Input	Output
smart	None	SMART dataset
predict	SMART dataset	Severity level
feedback	Prediction result	Sample weight
predictor_upd	SMART samples	New predictor

3.1 Monitor

*Monitor*s are implemented to gather the SMART attributes from the drives in SSM, which are required by *Predictor* to predict the health status of the drives. Also, this information is used by *Trainer* to improve the prediction model (build a new model). We employ multi-level *Collector*s to gather SMART samples. The *Collector*s at the bottom use *smart* to gather information directly from *Monitor*s, then send it to the upper level *Collector*s regularly after necessary

pre-processing such as adding drive identification information and data normalization. *Collector*s of each layer are selected according to the topological structure of the storage system. If one *Collector* fails, a new *Collector* will be elected to ensure the availability of SSM. Because the collection job is executed per hour, it has little impact on system services. Once the root *Collector* has accumulated enough samples, *Predictor* and *Trainer* will call the *smart* interface to get SMART dataset.

3.2 Predictor

In general, hard drives deteriorate gradually rather than suddenly. Most previous works simply output a binary classification result which can not accurately describe drives' health status. On the contrary, the Regression tree (RT) model proposed by [5] does not simply output good or failed, but a deterioration degree which can be regarded as the health degree. This model can achieve a detection rate above 96 % and therefore most data from soon-to-fail drives can be protected by SSM.

We want to deal with pre-warnings according to the level of urgency so that the limited system resources can be effectively used to migrating dangerous data. For this, a coarse gained severity level evaluation is enough. For example, considering a drive predicted to be failed 250 h later, we do not have to distinguish it from the other one with a predicted remaining life of 249 h, but make sure to prioritize the other one predicted to be failed within 150 h over it. Thus, we can define k severity levels by dividing the domain of RT output values into k ranges. In our prototype system, we use 5 equal length ranges. Level 5 represents the healthy status and level 1 means the most urgent status. *Predictor* is responsible for converting a continuous value output by the prediction model into a discrete severity level. To verify how good our method evaluate the level of urgency, we apply it to two real-world datasets. Dataset A is from [5] and dataset B is collected from another data center. Figures 2 and 3 show the predicted results of failed drives in the test sets, which are very close to the ground truth. There are respectively 95 % and 96 % predicted results that are exactly equal to or one level away from the ground truth on A and B. It is can concluded that our method assesses the level of urgency effectively and can be used to prioritize migration tasks (Table 2).

Table 2. Two dataset details.

Dateset name	Good drive	Failed drive (training/test)
A	22,790	434 (302/132)
B	98,060	243 (169/74)

The SMART attribute values of drives change over time, and failure reasons vary as the environment changes. As a result, *Predictor* will become ineffective as time goes by, and therefore *Trainer* updates the model periodically. It uses

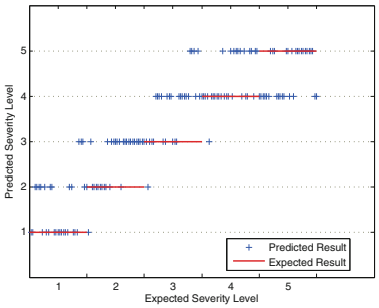

Fig. 2. Predicted severity level versus expected severity level in A.

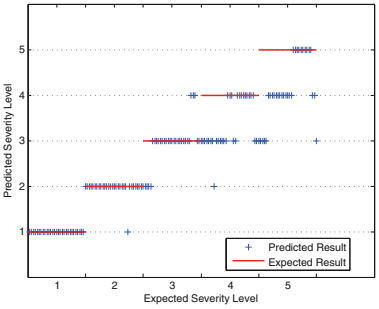

Fig. 3. Predicted severity level versus expected severity level in B.

the updating strategies proposed in [5] to build new models using the old and/or the new SMART samples. The *predictor_upd* interface is invoked to replace the old model by the new one, and then *Predictor* will use the new model so that the good prediction performance is maintained. Though systems always try to avoid missed alarms and false alarms, they are generally inevitable. When they arise unfortunately, the system will catch them and send the wrong predicted SMART samples to *Trainer* as the *feedback*. These samples will be used in model updating to improve the accuracy of the model.

3.3 Self-Scheduling Migration

When an alarm arises in the system, the data on the soon-to-fail drive should be effectively protected. A handling strategy can reasonably process multiple pre-warnings according to the current health status and the redundancy layout of the storage system. We design a self-scheduling migration algorithm, which migrates data on the soon-to-fail drive as soon as possible. The migration algorithm has three distinguished features. First, it uses a dynamic priority scheduling rather than the traditional first come first service discipline when migrating data. Different levels of priority (severity) possess different migration rates. We also have a good strategy to handle the priority changing with time. Second, we do not simply use the soon-to-fail drive as the migration source. The healthy drives containing the replicas of a dangerous data block may be selected as the source to slow the deterioration of the soon-to-fail drive. Third, we try to reduce the overhead of migration as much as possible. We measure the bandwidth required by the normal service and then set a migration bandwidth to ensure the migration has a low impact on the system. As Fig. 1 shows, we design three modules, priority based scheduling, multi-source migration and bandwidth allocation, in *Scheduler* to implement these features. The detailed self-scheduling migration algorithm is shown in Algorithm 1.

Algorithm 1. self-scheduling migration

Input: New soon-to-fail drives set W', current soon-to-fail drives set W
Output: none

```
1: Begin
2: W ← W ∪ W'
3: q ← NULL                                              ▷ q: priority queue
4: for each drive d in W do
5:     calculate score(d) using Eq. 1
6:     for each unmigrated block b on d do
7:         t_b ←create a migration task for b
8:         t_b.prio ←(score(d), dr(b))          ▷ dr: the number of dangerous replicas
9:         q.insert(t_b)
10:    end for
11: end for
12: calculate the total relative severity score by a reduction operation
13: for each drive d in W do
14:    calculate bt(d) using Eq. 2
15:    c(d) ← 0                                          ▷ c(d): bandwidth usage
16: end for
17: while q is not empty do
18:    q.deletemax(t_b)
19:    select the source drive S and the destination drive D for b
20:    if c(S) + m(b) < bt(S) then          ▷ m(b): bandwidth required to migrate b
21:        copy b from S to D                                ▷ perform migration
22:        update b's metadata and mark it as migrated
23:        if all blocks on the same drive d have been migrated then
24:            remove d from W
25:        end if
26:    end if
27: end while
```

Alarm Handling Mechanism. We introduce a drive alarm daemon (DAD) to implement self-scheduling migration. When *Predictor* reports pre-warnings (generally periodically), DAD will receive the alarms and perform the following steps as Algorithm 1 shows: firstly, scan all the unmigrated blocks on the soon-to-fail drives (line 4–12) and create a migration task for every block (line 7); secondly, allocate migration bandwidth for every drive (line 13–16); thirdly, select the source and destination drives for each migration task (line 19); and finally, perform migration tasks if there is enough migration bandwidth available and update metadata (line 20–26).

Data consistency is a critical problem. When a block is being migrated, users may update its content, where data inconsistency may occur. To address this issue, we adopt a fine-grained locking mechanism. While a data block is being migrated to a new drive, the system blocks write operations to it. Read operations are served as usual. Writes are unblocked after this migration task is accomplished. Since the locking granularity is just a block, it will not cause great impact on the system service.

Priority-Based Scheduling. It is not rare that multiple drive failures occur simultaneously in a large data center. Consequently, how to allocate reasonably migration bandwidth to multiple pre-warnings is the key problem. A reasonable strategy is to give more resources to the drives in higher severity levels. We introduce a *relative severity score* as the priority to control bandwidth allocation. It takes both severity level s and migration progress p into account. The migration progress is measured by the ratio of the number of migrated blocks to the total number of ones on the soon-to-fail drive. Drives with higher severity levels and lower migration ratios will be given a higher *relative severity score* which implies a larger share of migration bandwidth. The *relative severity score* of the ith drive is calculated as

$$score(i) = \frac{1 - p(i)}{s} \tag{1}$$

Line 5 in Algorithm 1 calculates the *relative severity score* for every block and line 8 uses the score as the priority of a block.

Multi-source Migration Algorithm. The fundamental difference between pre-warning handling and failure handling is that the soon-to-fail drives are still in operation. So an intuitive idea is to migrate data only from soon-to-fail drives. However, it will be bound to put more pressure on the soon-to-fail drives, which may accelerate their deterioration. SSM instead selects the source drive for a block D to be migrated from all of the drives having D's replicas, which fully uses the bandwidth and balances the load. More specifically, SSM selects the source drive by taking both load and health status of drives into account: firstly, prefer drives with lower load. Secondly, when the soon-to-fail drive deteriorates faster than ever, choose another source. Finally, when a source drive is being offline, choose another one. Then the destination drive is selected in the same way as normal replica creation except that the soon-to-fail drives are not considered as candidates. This ensures good system reliability. By using this multi-source strategy, we can achieve a better migration performance compared with traditional reactive systems.

Migration Bandwidth Allocation. To reduce the impact of migration on system service, we only allocate a proportion of available bandwidth to migration tasks and reserve the rest to serve normal service. Let α denote the percentage of allocated migration bandwidth in the total bandwidth and B denote the total bandwidth. Migration task is executed on the basis of what migration bandwidth is below αB. If the overload of migration tasks is below αB, they will be scheduled normally.

A drive with a higher *relative severity score* should be allocated a higher migration bandwidth on the basis of the same total migration bandwidth. We set a migration threshold $b(i)$ for every soon-to-fail drive i. The migration bandwidth is allocated according to Eq. 2. For a block b on the drive i, the migration task is performed if the migration rate of b's source drive S does not exceed its migration threshold $b(S)$.

$$b(i) = \alpha B * \frac{score(i)}{\sum_{i=1}^{n} score(i)} \tag{2}$$

where n is the total number of soon-to-fail drives and $score(i)$ is i's current relative severity score. Line 14 in Algorithm 1 calculates the migration threshold for every soon-to-fail drive.

3.4 Reliability Analysis

Related researches [5] show that accurate detection rates of prediction models can help increase the Mean Time To Data Loss (MTTDL) and thus improve the reliability of storage systems greatly. However, the building of prediction models is just the first step and far from enough. Our ultimate goal is to put these models into practice by guiding system's pre-warning process. Once the *Scheduler* receives pre-warnings, it will trigger the recover process to migrate the data on soon-to-fail drives in advance. As a result, by deploying the proactive fault tolerance mechanism, we can shorten the system reconstruction time and reduce the Mean Time To Repair (MTTR) as much as possible, which enhances the reliability of system significantly. On the other hand, given plenty of time for data migration, *Scheduler* can utilize system resources more efficiently. That is, while the system is heavy loaded, a low bandwidth will be limited in the migration process, whereas a high one can be adopted, which means side effects to normal read and write performance are minimized dramatically while compared with the original system without SSM.

4 Evaluation

In this section, we present the experimental results of SSM. We implement SSM as a modified instance of Sheepdog which is an open source project of a distributed storage system. Sheepdog provides a high available block level storage volumes and adopts a completely symmetrical architecture, which implies no central control node. The nodes and data blocks are addressed by Distributed Hash Table (DHT).

We set up a cluster comprising of 12 nodes to simulate a local part of a large scale distributed storage system. Since a Sheepdog system is completely symmetrical, experimental results on this local part can reflect the overall performance. Each machine runs CentOS 6.3 on a quad-core Intel(R) Xeon(TM) CPUs @ 2.80 GHz with 1 GB memory and a RAID-0 consisting of six 80 GB SATA disks. The machines are connected by Gigabit Ethernet. SSM in Sheepdog takes three replicas as the redundance strategy and uses the default 4 MB block size. Through the experiments, we try to show that (1) SSM is superior than reactive fault tolerance and (2) migration scheduling algorithm is effective in reducing the impact on system service.

4.1 Proactive Fault Tolerance Versus Reactive Fault Tolerance

An important advantage of SSM over traditional reactive fault tolerant technologies is that it can achieve good reliability while remaining the quality of service. In a reactive fault tolerant system, once a hard drive failure occurs, the system must recover it as soon as possible. This "best effort" strategy implies that the repair process will occupy a large part of the system resources which will affect the performance of users' read and write requests significantly. On the other hand, the system certainly can guarantee QoS by limiting the resources used by the repair process. However, that will lead to a much longer MTTR which is detrimental to the reliability. In other words, a reactive fault tolerant system cannot obtain both the reliability and QoS. In contrast, SSM can predict drive failures several days even several weeks in advance. Therefore, even though it only allocates a small share of disk and network bandwidth to the migration process, it still can complete the migration before the failure actually occurs. Since the state-of-art drive failure prediction method [5] maintains good prediction accuracy, few missed failures will not defer SSM from obtaining both good reliability and minimal impact on reading and writing service.

Table 3 compares degraded read and write throughput and MTTR (migration time) of SSM and RFT (reactive fault tolerance). SSM adopts multi-source migration and bandwidth limitation to evaluate its performance. We assume that 8 TB (typical per-node data volume in modern cloud storage systems) data needs to be recovered (migrated). As expected, RFT with "best effort" repair impacts normal read and write throughput (118 MB/s and 25 MB/s respectively) seriously although it guarantee a short MTTR. Limiting repair bandwidth (10 MB/s) in RFT (RFT(BL)) alleviates impact on QoS effectively but leads to an unacceptable MTTR. In contrast, SSM remains good QoS and achieves a short migration time (compared with the prediction time in advance). Moreover, we have enough space to further reduce migration bandwidth limitation to obtain better QoS because there still a wide gap between the migration time and the prediction time in advance.

Table 3. Reactive versus Proactive

Strategy	Read (MB/s)	Write (MB/s)	MTTR (hours)
RFT	80	15	14.56
RFT (BL)	100	22	58.24
SSM	110	23	66.58

4.2 Evaluating Migration Performance

We manually simulated a drive with 20 GB data that is going to fail to trigger the pre-warning and then the migration. The throughput of single-source migration (SS), about 28 MB/s, is 16 % slower than that of multi-source. Multi-source

migration (MS) dose not only improve the migration rate, but also achieves a more balanced load by diverting the pressure to multiple drives. Replicas are scattered well by the consistent hash algorithm.

Migration also affects the data access performance. We test the write performances with migration using the following steps. First, we start a sequential write job, then simulate a pre-warning to trigger the migration, and record the running state trace of the system when write and migration exist simultaneously. The test lasts 350 s which is long enough to reflect the correlativity between data access and migration. Also, we evaluate the read performance by a similar method. A pre-warning is triggered when the user requires a read service. Moreover, we implement MS to evaluate the read and write performance. In Fig. 4, write throughout is about 26 MB/s without any migration, while it is reduced by more than 80 % with migration not limiting the bandwidth. Quality of normal service drops down heavily when migration occupies lots of resources. We should make effects to decrease the drops. DHT constructs a ring and each node covers a part of the ring. Neighbors in the ring always have a higher similarity in data and resource. We allocate migration bandwidth based on a concept of locality. The ring are divided into many parts, and every part has their allocation bandwidth. In the part consisting of 12 nodes, 10 % (α) of the total bandwidth (B), about 10 MB, is allocated to the migration job. With bandwidth limitation (BL), the write throughout is only reduced by 13 %. From Fig. 5, read throughout is around 120 MB/s in normal condition through simulation of a disk drive failure. It is decreased by 8 %, namely about 110 MB/s in migration with bandwidth limitation. While without bandwidth limitation, read rate is as low as 95 MB/s. Since the system blocks write operations when the block is being migrated, write rate decreased more than that of read. The degradation of write and read performance are all acceptable for the system with bandwidth limitation.

We also explore read and write performance under four different strategies and the results are shown in Fig. 6. Except for the fault-free configuration which means no alarms nor migrations, SSM with BLMS has the best performance in all other cases, which has the minimal impact on the users' operations. Compared

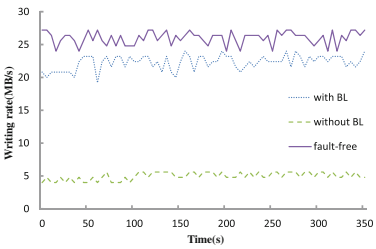

Fig. 4. Write rate in fault-free, bandwidth with limitation and without bandwidth limitation. Migration job has an great influence on write services. The writing operation with BL performs well.

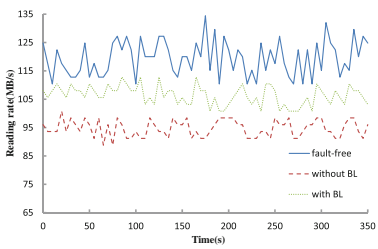

Fig. 5. Read rate in fault-free, bandwidth limitation and without bandwidth limitation conditions. Migration job has an light influence on read services. The reading operation with BL performs well.

with MS, the read throughput of BLSS is higher, which means BLSS can reduce the impact on system performance more effective than MS. When using MS without BLSS, there is a higher possibility that a write should wait for the lock acquired by the migration process as more drives participate in the migration. So the write performance of MS is worse than that of BLSS. In Subsects. 4.3 and 4.4, we all adopt BLMS and allocate 10 MB as the migration bandwidth.

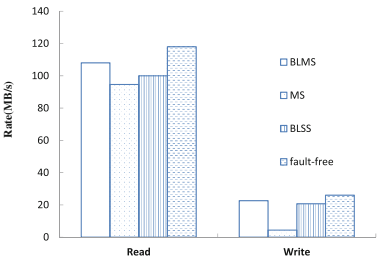

Fig. 6. Read rate and write rate of fault-free, BLSS, MS and BLMS.

4.3 Evaluating Priority-Based Scheduling

Our system migrates data according to the priority making drives with different severity levels treated differently. We simulate a process with multiple warnings in different severity levels. In Figs. 2 and 3, our proposed method that converts health degrees to severity levels is proven to be reasonable. We trigger a pre-warning with level 1 first, and the migration rate is about 30 MB/s. After 90 s, a level 2 alarm arrives with a lower priority. At the same time, the migration rate of the drive in level 1 decreases. Another pre-warning with the same severity level appears at the 150th second and its migration rate is almost equal to the first 2 warnings. At the 210th second, an alarm with severity level 4 was raised. From the 210th second to the 630th second, as is detailed in Fig. 7, the four migrations for the drive alarms run simultaneously. Migration for drives in level 1 has the maximal rate, the two warnings with level 2 have the middle rate and the minimum rate is held by the warning with the drive in level 4. The migration rate increases gradually on account of migration completing after 630 s.

4.4 Performance on Real-World Traces

SSM with BLMS has a very little influence on the normal operations in Sheepdog. All of the above experiments use synthesized workload. Now we choose three real traces (fileserver, webserver and netsfs) of Filebench to test our system. Figure 8 illustrates the throughout (Y-axis, IO/second) of the system. Compared with the fault-free condition, the throughput of fileserver is decreased by 15 %, that of webserver is decreased by 10 % and for the case of netsfs it decreases the least for only 1 %. SSM performs well with all the three cases.

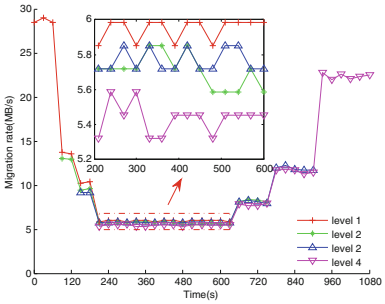

Fig. 7. The change of migration rates as four alarms with different severity levels are raised at different times.

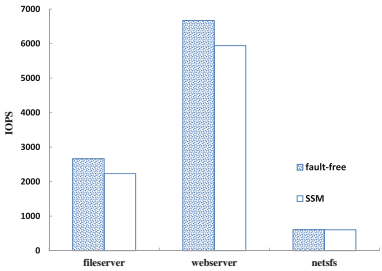

Fig. 8. IOPS of fault-free and pre-warning condition in the real workloads fileserver, webserver and netsfs.

5 Conclusion

This paper provides a proactive fault tolerance mechanism for a typical distributed system, Sheepdog. In our method, we migrate data on the soon-to-fail drives before disk failures really occur. By increasing the number of replicas for the blocks on these soon-to-fail drives, the reliability of storage system gets improved. When a failure happens, data reconstruction overhead will be significantly reduced. Different severity levels are proposed for the health degree. We introduce a *relative severity score* to evaluate the severity level. For a higher *relative severity score*, data on this drive will be migrated as soon as possible thus is given a high processing rate. On account of every block having several replicas, we take some conditions into consideration to choose a proper source. By combining fully multi-source migration and bandwidth limitation, SSM has a little influence on the original distributed system.

Adding SSM to the distributed system, we handle threatened data before failure, which will influence the original service. Migration rate is restricted to reduce the impact on system. Every soon-to-fail drive has their own migration mechanism to allocate migration bandwidth. As a result, the system only causes respectively 8 % and 13 % performance drops on read and write operations. Compared with traditional reactive fault tolerance, SSM significantly improves system reliability and availability.

Acknowledgments. This work is partially supported by NSF of China (grant numbers: 61373018, 11301288), Program for New Century Excellent Talents in University (grant number: NCET130301) and the Fundamental Research Funds for the Central Universities (grant number: 65141021).

References

1. Vishwanath, K.V., Nagappan, N.: Characterizing cloud computing hardware reliability. In: Proceedings of the 1st ACM Symposium on Cloud Computing, pp. 193–204. ACM (2010)
2. Bairavasundaram, L.N., Goodson, G.R., Pasupathy, S., Schindler, J.: An analysis of latent sector errors in disk drives. ACM SIGMETRICS Perform. Eval. Rev. **35**, 289–300 (2007)
3. Bairavasundaram, L.N., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H., Goodson, G.R., Schroeder, B.: An analysis of data corruption in the storage stack. ACM Trans. Storage (TOS) **4**(3), 8 (2008)
4. Allen, B.: Monitoring hard disks with smart. Linux J. (117), 74–77 (2004)
5. Li, J., Ji, X., Zhu, B., Wang, G., Liu, X.: Hard drive failure prediction using classication and regression trees. In: DSN (2014)
6. Qin, A., Hu, D., Liu, J., Yang, W., Tan, D.: Fatman: cost-saving and reliable archival storage based on volunteer resources. Proc. VLDB Endow. **7**(13), 1748–1753 (2014)
7. Wu, S., Jiang, H., Mao, B.: Proactive data migration for improved storage availability in large-scale data centers (2014)
8. Patterson, D.A., Gibson, G., Katz, R.H.: A case for redundant arrays of inexpensive disks (RAID) **17**(3), 109–116 (1988)
9. Blaum, M., Brady, J., Bruck, J., Menon, J.: Evenodd: an effcient scheme for tolerating double disk failures in raid architectures. IEEE Trans. Comput. **44**(2), 192–202 (1995)
10. Cidon, A., Rumble, S.M., Stutsman, R., Katti, S., Ousterhout, J.K., Rosenblum, M.: Copysets: reducing the frequency of data loss in cloud storage. In: USENIX Annual Technical Conference, pp. 37–48. Citeseer (2013)
11. Ford, D., Labelle, F., Popovici, F.I., Stokely, M., Truong, V.A., Barroso, L., Grimes, C., Quinlan, S.: Availability in globally distributed storage systems. In: OSDI, pp. 61–74 (2010)
12. Hafner, J.L.: Weaver codes: highly fault tolerant erasure codes for storage systems. In: FAST, vol. 5, pp. 16–16 (2005)
13. Papailiopoulos, D.S., Luo, J., Dimakis, A.G., Huang, C., Li, J.: Simple regenerating codes: network coding for cloud storage. In: INFOCOM, 2012 Proceedings IEEE, pp. 2801–2805. IEEE (2012)
14. Murray, J.F., Hughes, G.F., Kreutz-Delgado, K.: Machine learning methods for predicting failures in hard drives: a multiple-instance application. J. Mach. Learn. Res. **6**, 783–816 (2005)
15. Ma, A., Douglis, F., Lu, G., Sawyer, D., Chandra, S., Hsu, W.: Raidshield: characterizing, monitoring, and proactively protecting against disk failures. In: Proceedings of the 13th USENIX Conference on File and Storage Technologies, pp. 241–256. USENIX Association (2015)