A Simulation Result of Replicating Data with Another Layout for Reducing Media Exchange of Cold Storage

Satoshi Iwata
FUJITSU LABORATORIES LTD.[1]

Kensuke Shiozawa
FUJITSU LABORATORIES LTD.

Abstract

Cold storage devices such as tape and optical discs are a good solution for reducing the total cost of ownership for storing data. However, there is a drawback in that media and drives are separated, and placing media into drives when accessing data needs a few minutes of time. Though placing correlated data together in the same medium reduces media exchange, multi-dimensional searches disrupt it. We propose two approaches which replicate data and place them in different layout for solving the problem. By concentrating on relative latency reduction or utilizing replicas originally generated for avoiding data loss, our method achieves high latency reduction with restricted capacity efficiency loss. A simulation result shows 31% average relative latency reduction with capacity efficiency remaining at 91%.

1 Introduction

Reducing the cost for storage is an important task, since the amount of data is continuously and sharply increasing. One solution for achieving this goal is to migrate less frequently accessed data appropriately to cold storage devices such as tape or optical discs, which offer a lower performance but are cheaper. The lower cost of those systems is due not only to the price of the devices themselves but also to reduced electricity usage and long media life. Media themselves need no electricity unless they are accessed. Media life is 30 years for a tape and 50 years for an optical disc, so the migration cycle for keeping data is longer compared to a hard disk, of which the life is around five years.

One barrier against installing such cold devices is a large latency incurred by a media exchange. In contrast to all-in-one devices such as a hard disk drive (HDD) or a solid state drive (SSD), a cold storage medium is separated from a drive and has to be set into a drive to allow access. It usually requires a few minutes (c.f., two minutes in some libraries). Cold storage devices are becoming popularly used for active archiving and thus, reducing media exchange is a major issue. Analysis of 100PB storage systems showed that media exchange is the norm with nine loadings every minute on average [6].

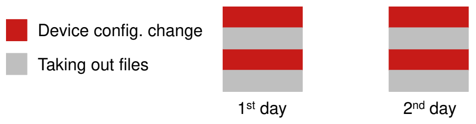

Figure 1: **A Layout Suitable for Retrieval without Any Filtering.** One square indicates a medium.

Reducing media exchange can be achieved by placing correlated data in the same medium. We would like to show a use case for storing log data. Log data is a good example of using cold storage, since it becomes enormous in a big system and can be kept for a long time as evidence. Moreover, it is rare for all the logs to be accessed frequently. In most cases, logs are retrieved focusing on time. One case might be discovering a root cause of a failure by checking logs around the time of occurrence. Thus, storing these data in the order of generation as in Fig. 1 would be best. In this example, two types of log interleave as they are generated.

However, there can be multiple ways to mine the same logs. When searching for data focusing on either type, the layout shown in Fig. 2 is the preferred option. An example might be searching for a malicious employee who leaked confidential information by checking only `Taking out files` action taken in this particular week retrospectively. With the layout in Fig. 1, media exchange is required to collect two days' logs for each action but it is not required with the layout shown in Fig. 2. However, in contrast, the layout in Fig. 1 works better for collecting one day's logs for all actions. This log scenario is just one example of multiple mining. Another is weather information used for forecast searched in both temporal and spatial manner.

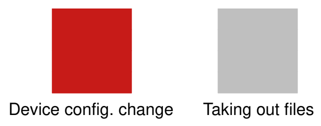

Figure 2: **A Layout Suitable for Retrieval with Filters Regarding Actions.**

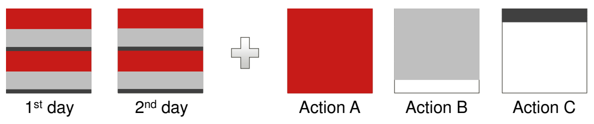

Figure 3: **Layout Replicating Large and Small Actions Respectively.**

Our basic idea to meet both requirements is to replicate each log, maintain both layouts in Fig. 1 and Fig. 2, and choose an appropriate layout depending on each query. However, keeping both layouts reduces capacity efficiency by half. To avoid this deficiency, we propose to focus on a *relative* latency reduction. By applying this notion, latency is reduced by 31% on average, with capacity efficiency remaining at 91% with our sample logs. Moreover, we are proposing a placement policy of replicas originally generated for improving reliability. Utilizing existing replicas avoids capacity efficiency loss for accelerated processing.

The rest of the paper is organized as follows. Sec. 2 describes sample log data and the search queries we are focusing on. Our approaches are explained in Sec. 3 and the results of simulations are shown in Sec. 4. Related works are referred to in Sec. 5 and, finally, we summarize this paper in Sec. 6.

2 Sample Log Data

We use logs of users' actions collected on their PCs in an anonymized format throughout this paper. An example log is as follows.

```
WIN-HOST,2014/10/6 15:34,user1,G01,Run
  application,Legal, ,Ran [iexplore]
```

Each log includes time (`2014/10/6 15:34`), user ID (`user1`), action (`Run application`), and so on.

These logs are predominantly mined in two ways. The first type of query needs all actions. An example is as follows and shows numbers of actions.

`stats count by "action"`

The other is a type focusing on each action as

`"action"="Device configuration change" | timechart count by "Legality"` or

`"action"="Taking out files" | timechart count by "Legality"`.

So, a layout shown in Fig. 1 matches the first type and another layout shown in Fig. 2 corresponds to the second type.

We also show percentages of logs in size depending on each action. There are 11 actions and each log includes exactly one action. There are six filters regarding actions and these are `Log off OR Log on`: 1.0%, `Print out`: 2.6%, `Taking out files`: 2.8%, `Device config. change`: 3.2%, `Run application`: 17.5%, and `File manipulation`: 20.8%. `Log off` and `Log on` are separate actions, but are always checked together.

3 Our Approach

3.1 Considering Relative Latency Reduction

Let us imagine that we are collecting action-oriented logs of two consecutive days. Assuming that the quantity of logs are different for each action as in Sec. 2, if the quantity is 50% of total logs, a medium dedicated to this particular action becomes an `Action A` medium in Fig. 3. We can reduce one media exchange, resulting in two minutes' latency reduction. However, this is relatively small. Bearing in mind that reading an entire medium requires approximately 333 minutes with an LTO Ultrium7 tape and about 93 minutes with a blu-ray disc, the relative time reductions are 0.6% and 2.1% respectively. Though capacities are different between those devices, we concentrate on effects of data placement policy. Comparison between devices are out of the scope of this paper.

On the contrary, if the fraction is low and 1% as `Action C` in Fig. 3, latency decreases from 10.7 minutes to 8.7 minutes with a tape and from 5.9 minutes to 3.9 minutes with a blu-ray disc. Thus the reductions become 18.7% and 33.9%, respectively. Computation formulae can be found in Sec. 4.1.

We propose to replicate small fraction data such as `Log off OR Log on` rather than `File manipulation`. We believe that two minutes' reduction from 11 minutes is much more valuable compared to a reduction from five and a half hours. Moreover, using this method, large capacity efficiency loss can be avoided. Replicating `File manipulation` reduces capacity efficiency by 17%, in contrast to `Log off OR Log on` which reduces it by just 1%. In this paper, we assume that the frequencies of all search queries are the same, since we do not have the information for our sample data. Where the frequencies vary, prioritizing replicating larger amount actions can be beneficial.

It may be that less-utilized media are required to store small-fraction actions as `Action C` in Fig. 3. However, more logs of `C` are produced as time goes on and occupy the remaining portion, so this does not cause par-

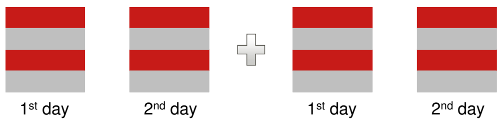

Figure 4: **Storing Replica with the Same Layout.**

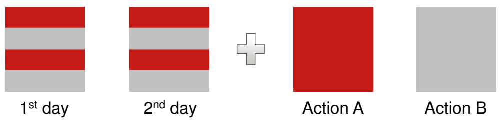

Figure 5: **Storing Replica with a Different Layout.**

ticularly significant ineffectiveness. The number of replicas needed matches the number of dimensions we need to support in our queries. While this reduces the capacity efficiency as the number of dimensions increases, our method can be applied to data mined over two dimensions. For example, if we add two replicas, we can prepare three medium for different search queries; time order, action specific, and user specific.

3.2 Using Replicas for Avoiding Data Loss

If a user decides to replicate logs to reduce the probability of data loss, our approach can use them for reducing media exchange also. We often replicate media as a whole as in Fig. 4. Our approach replicates data but stores them in a different layout as in Fig. 5; one for retrieving all actions and the other for filtered searches. It does not reduce capacity efficiency, since the data are not added for our purpose, but our goal is met. The number of search dimensions can be increased as the number of replicas increases. Moreover, we can add replicas of small fraction actions to adapt to dimensions which exceeds the number of prepared replicas.

However, there is a side effect in that availability is slightly reduced. When two media fail, data loss happens when they share the same content. With our layout, the probability of two media sharing the same content becomes double. In Fig. 4, 1st day's medium shares the same data only with the other 1st day's medium. In contrast, in Fig. 5, 1st day's medium shares the same data with both Action A's and B's media. Although double, we believe this is not a significant problem, since the probability of two media failing at the same time is quite small and thus the doubled amount does not affect a lot. Of course, if any decrease in data loss probability is unacceptable, our layout should not be applied.

At the end of this section, we explain that frequent media exchange is not required for replicating an arriving log into multiple media. We believe that it is natural to prepare the primary tier for storing recent logs and demote them as they become old. So, we can minimize

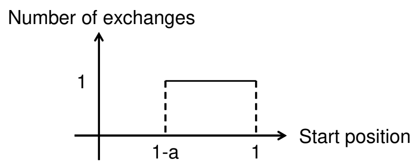

Figure 6: **Number of Media Exchanges Depending on Start Position.**

media exchange for write requests while read requests are few. Moreover, to implement our method, management of index to the media is required. We believe it is not a big deal, since the managed data is not large; a bit indicating time order or action specific, start time, end time, and action type if it is an action specific media.

4 Simulation

4.1 Conditions

First we would like to explain how the amount of media exchange is modeled depending on the time period over which we are collecting. Let us imagine that each medium holds one day's logs as in Fig. 1. Even though the collection amount is the same (e.g., a half day), the number of media exchanges depends on which period is required. Logs collected from 6 A.M. to 6 P.M. on the first day do not require media exchange. However, if logs are collected from 6 P.M. on the first day to 6 A.M. on the second day, media exchange is required.

We consider three different models; mean, minimum, and maximum. Mean is modeled as $1+x$, minimum as $1+\lfloor x \rfloor$, and maximum as $1+\lceil x \rceil$. x equals the amount of data reading, which is normalized with media size. Initially, 1 media exchange is required for each model to insert the first medium. The remainder constitutes exchanges required while reading. Due to space constraints, we focus on mean in this paper.

In a mean model, let us replace x with $n+a$. n is an integer part and a is a decimal part. With regard to the decimal part, the number of media exchanges alters depending on the start position. We explain this using Fig. 6. Until the start position reaches $1-a$, media exchange is not required. If the start position exceeds $1-a$, one media exchange is required. Assuming that the start times are equally distributed, the mean media exchange becomes $1-(1-a)=a$. Moreover, reading n data, which is a multiple of media size, requires n media exchanges on average. So, the required exchange becomes $n+a=x$.

In this paper, for simplicity, we assume that the time for reading data increases proportionally to the data amount. In other words, we exclude seek time from la-

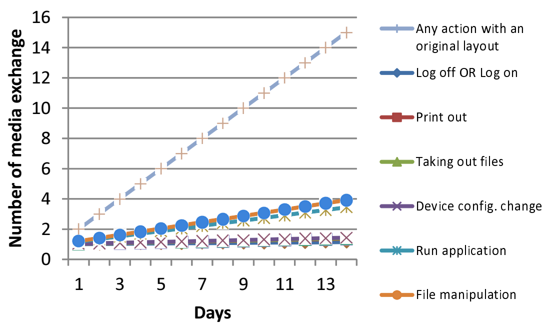

Figure 7: **Mean Number of Media Exchanges.**

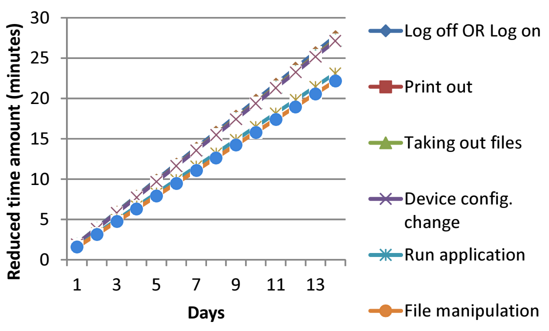

Figure 8: **Reduced Mean Latencies in Minutes.**

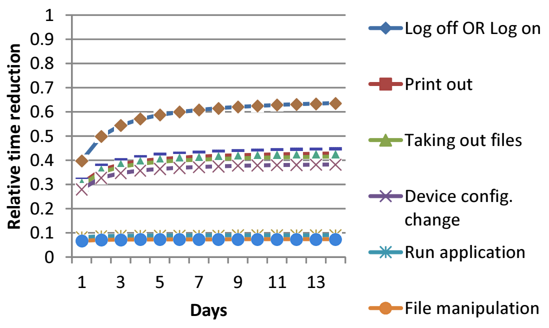

Figure 9: **Relative Latency Reduction.**

tency. With a blu-ray disc, reading all the data requires around 93 minutes, and so we assume that reading 50% requires 46.5 minutes and reading 5% requires 4.65 minutes. Though it may affect the results, it is not favorable to our approach. Random accesses occur with an original interleaving layout, but this does not occur with our action-oriented layout. The retrieving action becomes as follows in the action-oriented layout, since we assume that logs are retrieved focusing on consecutive time periods and are stored in that order: a read head is placed on the first log and then data is read until the end of the period.

We assume that indexes of logs are appropriately prepared. If there are no indexes, reading of data is required to position the read head at the start point. We assume that indexes direct the head to the start position. We believe blu-ray discs are more suitable than tapes for accelerated processing, since the positioning latency is much less. Thus, we focus on that device in this simulation, though our approach can be applied to both devices.

Using the above assumptions, we simulate five metrics; number of media exchanges, latency, absolute latency reduction, relative latency reduction, and capacity efficiency. Latency is the amount of time required for reading and media exchange. Relative latency reduction can be calculated by dividing the reduced time amount by the original latency.

4.2 Results

First of all, we show the mean number of media exchanges. In this paper, to facilitate understanding, we assume that, in an original layout, one media holds a full-day's interleaving logs as in Fig. 1. The number of media exchanges depending on the retrieving period with an original layout is depicted at the top in Fig. 7. The number of exchanges does not alter depending on each action, since each medium holds a full day's logs of all the actions.

With action-oriented replicas as in Fig. 2, the mean number of media exchanges is reduced as in Fig. 7. The numbers are different among the actions, because of the difference in log amounts; the fewer an action's log amount is, the more days a medium can store.

Reduced mean latencies are as per Fig. 8. This shows a monotonic increase, since more media exchanges can be reduced as the access period increases. Moreover, the effect is greater with fewer logs, since more days can be stored in one medium.

Relative latency reduction becomes as per Fig. 9. As we explained before, fewer logs see a greater relative reduction, though absolute reduction does not differ a lot. This is because the latency for reading data is much less with fewer logs.

Finally, we show which configurations are more balanced when considering both latency reduction and capacity efficiency in Fig. 10. The horizontal axis shows which actions are replicated in a cumulative manner and the vertical axis shows the ratio of both latency reduction and capacity efficiency. We show three scenarios; collecting for one day, seven days, and 14 days. Latency reduction depends on how long we are collecting logs. As you can see from Fig. 10, the reduction increases as the collecting period becomes longer as we have explained.

The left-most column shows an original layout without any replicas. Its capacity efficiency is one but there is no latency reduction. Moving to the right and repli-

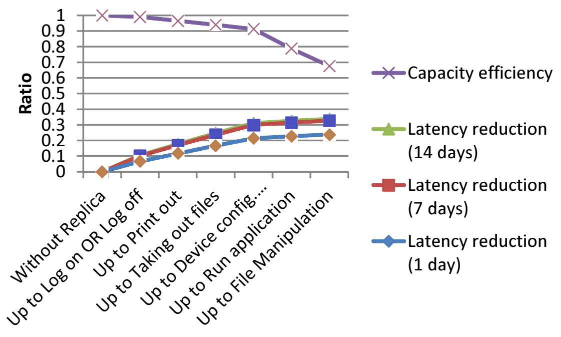

Figure 10: **Finding a Balanced Layout.**

cating more actions, efficiency decreases and latency reduction increases. In our sample logs, replicating logs up to `Device configuration change` would be the best option. To collect logs of 14 days, this gives a 31% average latency reduction among all six queries with filtering, and retains a capacity efficiency of more than 91%.

At the end of representing simulation results, we note that our method described in Sec. 3.2 gives about 34% relative latency reduction on average without any capacity efficiency loss for accelerated processing.

5 Related Work

Two methods have already been proposed for reducing media exchange. However, neither of them takes into account multi-dimensional searching. Adams et al. [2] propose an ad-hoc approach for reducing media exchange such as placing the same user's data in one medium. DedupT [5] is a method which avoids media exchange for recovering deduplicated data, by avoiding deduplication among media.

Multi-dimensional searches in tape systems are proposed [4]. Their approach revolves around clustering depending on past retrieval actions without adding any replicas. So, one datum cannot belong to multiple clusters and exchange reduction cannot always be achieved. Our approach fulfills the goal by producing replicas. Capacity efficiency and retrieval performance are tradeoffs.

Pelican [3] is an HDD-based cold storage system which restricts the number of spinning HDDs in parallel. Our method can be applied to Pelican to place sequentially-accessed data together in the same HDD and avoid frequent switches of spinning disks.

Some people might think that our approach's basic idea is similar to column-oriented database [1], which reduces latency by replicating data in another layout. The main difference between them is our goal is reducing media exchange and thus, is able to add a notion of relative latency reduction.

6 Summary

We proposed two approaches to reduce media exchange in cold storage devices. To enable multi-dimensional searching in the same logs, our approaches replicate logs in a different layout from the original one. One approach involves replicating fewer logs which results in a considerable relative latency reduction and low capacity efficiency loss. The other prevents any capacity efficiency loss by placing replicas originally produced for avoiding data loss in a different layout.

With the collected sample logs, our method achieved a 31% relative latency reduction on average, with capacity efficiency remaining at more than 91%. When replicas are already prepared for avoiding data loss, our method achieved about 34% relative latency reduction on average without any capacity loss. We are planning to confirm the effect of our method with real hardware such as optical discs, which should give more realistic results including seek latency.

7 Acknowledgments

We would like to thank Mr. Kazuhiko Nakamura, Mr. Satoshi Ogata, and other members of Splunk reseller at the FUJITSU SOCIAL SCIENCE LABORATORY LTD. for providing us with sample logs collected at their site. We also would like to thank our shepherd Nohhyun Park for giving us valuable comments for the final paper.

References

[1] Abadi, D., et al. The design and implementation of modern column-oriented database systems. *Foundations and Trends in Databases 5*, 3 (2012).

[2] Adams, I. F., et al. Usage behavior of a large-scale scientific archive. In *Proc. of the Conf. for High Performance Computing, Networking, Storage and Analysis* (2012), IEEE and ACM.

[3] Balakrishnan, S., et al. Pelican: A building block for exascale cold data storage. In *Proc. of the Symp. on Operating Systems Design and Implementation* (2014), USENIX.

[4] Chen, L. T., et al. Efficient organization and access of multi-dimensional datasets on tertiary storage systems. *Information Systems 20*, 2 (1995).

[5] Gharaibeh, A., et al. DedupT: Deduplication for tape systems. In *Proc. of the Conf. on Massive Storage Systems and Technology* (2014), IEEE.

[6] Grawinkel, M., et al. Analysis of the ECMWF storage landscape. In *Proc. of the Conf. on File and Storage Technologies* (2015), USENIX.

Notes

[1] I am temporary transferred to NIFTY Corporation, which is another subsidiary of FUJITSU.