Exploiting User Activeness for Data Retention in HPC Systems

Wei Zhang
Texas Tech University
Lubbock, Texas, USA
X-Spirit.zhang@ttu.edu

Suren Byna
Lawrence Berkeley National Laboratory
Berkeley, California, USA
sbyna@lbl.gov

Hyogi Sim
Virginia Tech
Blacksburg, Virginia, USA
hyogi@vt.edu

Sangkeun Lee
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
lees4@ornl.gov

Sudharshan Vazhkudai
Micron Technology
Austin, Texas, USA
svazhkudai@micron.com

Yong Chen
Texas Tech University
Lubbock, Texas, USA
yong.chen@ttu.edu

ABSTRACT

HPC systems typically rely on the fixed-lifetime (FLT) data retention strategy, which only considers temporal locality of data accesses to parallel file systems. However, our extensive analysis based on the leadership-class HPC system traces suggests that the FLT approach often fails to capture the dynamics in users' behavior and leads to undesired data purge. In this study, we propose an activeness-based data retention (ActiveDR) solution, which advocates considering the data retention approach from a holistic activeness-based perspective. By evaluating the frequency and impact of users' activities, ActiveDR prioritizes the file purge process for inactive users and rewards active users with extended file lifetime on parallel storage. Our extensive evaluations based on the traces of the prior Titan supercomputer show that, when reaching the same purge target, ActiveDR achieves up to 37% file miss reduction as compared to the current FLT retention methodology.

KEYWORDS

Storage tiering, storage resource management, data retention, user behavior, data management, purge policy

ACM Reference Format:
Wei Zhang, Suren Byna, Hyogi Sim, Sangkeun Lee, Sudharshan Vazhkudai, and Yong Chen. 2021. Exploiting User Activeness for Data Retention in HPC Systems. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21), November 14–19, 2021, St. Louis, MO, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3458817.3476201

1 INTRODUCTION

HPC systems usually offer a large, shared scratch space - a file system that provides high performance and parallel file access to applications. Although this scratch space is not designed to store data permanently, many users use the space as a normal file system where they store their data files without any plan on releasing the space voluntarily. On the other hand, as the upgrade process of an HPC storage system usually takes a vast amount of investment, time and effort [7, 15, 34, 47], the total capacity of the scratch space tends to remain fixed for a considerably long time after the deployment of a system. Additionally, HPC applications constantly generate a tremendous amount of data [1, 2, 9, 14, 21–23, 35], making it necessary to manage storage resource effectively [11, 13, 18, 19, 38, 48] with, in particular, "data retention" – a process of retaining useful files and purging unimportant files to improve the utilization of storage space.

Over the years, various data retention methodologies have been proposed with multiple transitions between data retention criteria. Existing data retention methodologies include the fixed lifetime strategy (FLT) where files are purged according to a fixed definition of file lifetime, the value-based approach where files are purged according to various definitions on file value, and the "scratch-as-a-cache" approach where the scratch space is used as a cache for job executions. However, the value-based approach remains conceptual because the inconsistent definitions of file value among its variants compromise the applicability of this approach. Also, the scratch-as-a-cache approach is problematic because it requires intensive file loading and off-loading at the beginning and the end of a job execution and hence can significantly increase the job execution time, making it more complicated to craft the job scheduling algorithm. In fact, to the best of our knowledge, there is almost no sign of the value-based and the scratch-as-a-cache approaches in practice.

Today, the fixed-lifetime (FLT) data retention strategy is still the dominating data retention solution being used in the vast majority of HPC systems, while other approaches are rarely found in real practice. Therefore, we take the FLT approach as the foundation of our discussion in this study. In Table 1, we show several examples of FLT at different facilities. With this strategy, the data retention process can be automated by periodically scanning and purging the stale files that have exceeded their lifetime. As a result, the files that are recently accessed within the specified file lifetime will be retained. The underlying assumption of this retention solution is that the most recently accessed files will be more likely accessed again in the near future. In other words, the FLT data retention only considers the temporal locality of data files in scratch space.

While it is widely recognized that the temporal locality is of great importance in storage management, we observe that the scratch

SC '21, November 14–19, 2021, St. Louis, MO, USA

ACM ISBN 978-1-4503-8442-1/21/11…$15.00
https://doi.org/10.1145/3458817.3476201

Table 1: Data retention solution at various HPC facilities

HPC Facility	SCRATCH Retention Solution
NCAR [27]	Purge any 120-day old
OLCF [31]	Purge any 90-day old
TACC [40]	Purge any 30-day old
NERSC [28]	Purge any 12-week old

space management needs to consider more than the temporal locality. In fact, the file access pattern in the scratch space is often influenced by users' behavior. For example, in a project execution that drives multiple runs of HPC applications, due to various factors such as temporary administrative suspension of the project or task switching in users' workflow, the users may not be able to continuously work on their data files. Therefore, it is very common that users may leave their data files untouched for quite a long time and then come back to access these files. Thus, the FLT data retention strategy often leads to undesired file misses for users. Since many datasets contain a large number of files and the size of each data file can be large too, the procedures of collecting data files are usually complicated and hard to repeat. Therefore, recovering these files is not only expensive but also time-consuming, frustrating and, sometimes, even impossible. In addition, knowing the fixed file lifetime specified by the FLT, some users can game the system by "touching" their files periodically [26], as long as the lifetime of their files is "renewed" before fixed-lifetime data retention wipes out these files. This trick can lead to underutilized storage space if the users only reserve the files but rarely use them.

In summary, the temporal locality alone is insufficient for evaluating the file access pattern in the scratch space. Therefore, the FLT data retention is often unable to capture the dynamics of users' behavior with a fixed file lifetime setting and hence leads to numerous problems such as undesired file misses, unnecessary hassle for expensive file re-transmission process, and underutilized scratch space. In this study, we rethink the data retention problem from a holistic perspective that focuses on evaluating activeness of users that use a HPC system, access their files, etc. and activeness of users producing outcomes (i.e., completing jobs and tasks, producing analysis results, publications using a data set, etc.). Based on such a perspective on users' activeness, we propose **ActiveDR**, an **activeness-based data retention strategy** that considers activeness of users at the core of its design. ActiveDR has an efficient activeness evaluation algorithm and measures the frequency and the impact of user activities within a specified number of periods. Then it ranks the activeness of each user during these periods. ActiveDR categorizes users by the activeness and purges files based on that. It rewards active users with extended file lifetime based on their activeness rank. The retrospective file purging mechanism of ActiveDR ensures that the specified purge target will be guaranteed while prioritizing purging inactive users' files. Overall, with a user-centric view, ActiveDR is a unique and effective data retention solution that, to the best of our knowledge, promotes active and fruitful use of the HPC system.

We have evaluated the efficacy of ActiveDR using two years of system traces from Titan supercomputer and its Spider II storage system. Our evaluation result demonstrates that, when reaching the same purge target, ActiveDR effectively reduces up to 37% of file misses by retaining up to 213.47% more data for active users, as compared to the fixed-lifetime data retention method. Also, with ActiveDR, up to 95% active users are exempt from the file misses by the data purge operations. Furthermore, the ActiveDR takes less than 500MB memory footprint when evaluating the system traces and its activeness evaluation process finishes rapidly, within one second. Overall, the ActiveDR takes about one hour to finish the entire data retention process for over 935 million files. Our prototype release of ActiveDR can be found from `https://doi.org/10.5281/zenodo.5168853`.

The rest of the paper is organized as follows. In Section 2, we introduce state-of-the-art data retention strategies and discuss their drawbacks in detail. We then discuss the design principle of ActiveDR and detail its design in Section 3. After presenting the experimental result of our evaluation in Section 4, we conclude our work in Section 6.

2 RELATED WORK

In numerous studies on storage resource management [38, 39], the data retention was performed by defining a fixed lifetime of the files and then monitoring the file access time. This fixed-lifetime (FLT) data retention strategy only relies on the temporal properties of the files. Backed by the temporal locality theory, the FLT approach is widely accepted as it is believed that a data file will not be accessed again if it has not been accessed for a long time. In addition, the FLT approach is simple and easy to be implemented; hence FLT is used in most HPC systems.

Several studies proposed data value-based approaches [43, 48], which include more file attributes into the data retention criteria, such as the file type, file size, file age, file access frequency or a combination of them. The value-based approaches then led to a series of studies on finding the true definition of the file value [4–6, 8, 10, 17, 25, 37, 39, 41, 42, 45, 49]. However, as discussed by Attard et al. [3], "there is no consensus on the definition of data value", and the methodology of assessing or quantifying the value of data is currently incomplete. This drawback leads to limited interoperability of value-based approaches as their data value specifications remain different. Consequently, value-based approaches may introduce additional complexity in finding the most appropriate definition of file value for a particular HPC storage system and hence its practicality is substantially compromised. Therefore, we exclude the value-based approach from our following discussion not only because it is impractical but also because it would not be objective if we pick any of its variants for further discussion.

Monti et al. [26] proposed a "scratch-as-a-cache" data retention solution where an HPC scratch space is considered as a cache for jobs running on HPC systems. In this solution, a data file can only stay in a given scratch space if an application is using it. While this solution may be helpful in restoring the scratch space in a timely manner, it may cause frequent loading of files from an archive and purging operations that are time-consuming. Also, frequent data loading and off-loading procedure on large files can impose a heavy I/O burden on the storage system as well. Hence we exclude this approach from our following discussion as well since it may significantly lengthen the execution time of an application (or even the entire workflow) and may introduce unnecessary performance challenge to the storage system.

While both value-based approach and scratch-as-a-cache approach are rarely used in real practice due to their limited practicality, in a majority of HPC storage systems today, the most widely used data retention solution is still the fixed-lifetime (FLT) data retention methodology. Essentially, the FLT method retains the files that are accessed within a specified amount of time called “file lifetime”. The underlying assumption of FLT is that the most recently accessed files will be accessed again in the near future (or within the file lifetime, to be specific).

However, users may not access files in the specified lifetime that the FLT data retention methodology expects. Therefore, the FLT is often unable to capture the dynamics in users’ behavior and hence results in undesired file removal. In some cases, the users may process data files through multiple iterations and they need to access different sets of files back and forth. While some data processing iterations may last for months and only involve a particular set of the files, other files that are useful to future data processing iterations may be purged by the FLT retention process. Thus, the users may have to reload or restore those files in order to proceed with further data processing iterations. In some other cases, the users may be temporarily distracted from their data processing tasks due to unanticipated interruptions. For example, some users may find it necessary to temporarily hold their project, to conduct additional field studies or to collect additional data right after storing some data in scratch space. If the field study or the additional data collection process takes longer than the specified file lifetime, the FLT approach will purge the data files that are previously loaded, causing file misses for the users when they access their data files.

To verify how frequently and significantly the FLT method may introduce file misses to users, by courtesy of Oak Ridge Leadership Computing Facility (OLCF), we ran an emulation on the job and system traces during 2015 and 2016 at OLCF. We formulated a virtual file system by collecting the paths of all accessed files from the command lines in the job submission traces. We emulated the file accesses in 2016 while applying the FLT method with 90-day file lifetime and 7-day purge trigger interval. From the result plotted in Figure 1, we can see that, during the 366 days in 2016, the file miss ratio fluctuates randomly around 5%, between the lowest 0% and the highest 95.66%. For over 120 days, the file miss ratio is between 1% and 5%, and the file miss ratio runs between 5% and 30% for 99 days. Although the file miss ratio exceeds 30% for only 39 days, the result shows that the users may intermittently suffer from 5%-100% daily data access interruption during 138 days, almost half of the entire year. As there is no mechanism for users to recover their missing files automatically, it can take hours to days for the users to recover their data by either re-transmission or re-generation of the data, which will cause significant amount of network traffic, computing cycles and even project delay.

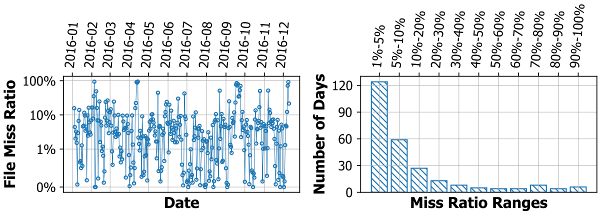


Figure 1: File misses introduced by FLT retention method

In fact, the scratch space is typically built to serve the short-term high-performance parallel accesses from batch jobs [32]. If the users need to keep their files for a long-term data processing task, they often need to manually manage their data files, migrating them to archival storage and loading them back to scratch space when needed, which is time-consuming and inconvenient. According to the observation reported in a prior study [26], some users may even game the FLT by “touching” their data files periodically to avoid undesired file purge against the temporarily unused data files. Such practice can lead to underutilized storage space. We observe that the activeness of different users may vary significantly, and the FLT data retention methodology ignores such variations. For example, some users working on data analytic workload, require an increasing amount of scratch space from time to time, while other users may only access their data once in a while. Therefore, it is time to devise a novel data retention solution which avoids undesired file purge as much as possible for active users, boosts the overall utilization of the storage space, and encourages fruitful usage of HPC systems.

3 ACTIVENESS-BASED DATA RETENTION

Being aware of the limitations we observed from the FLT data retention method, we envision that a better data retention solution should consider the file availability to the active users as well as the overall utilization of HPC systems towards fruitful outcomes. Therefore, the activeness of users should be well considered in the data retention solution. In addition, such a data retention solution should be able to integrate with the current data management practices in an automated fashion and does not require extensive tuning or training efforts from system administrators. Meanwhile, the solution should be efficient so that the purge decision can be made in a timely manner without taking a significant amount of memory. In this study, we introduce a novel activeness-based data retention solution, or ActiveDR in short, to address the limitations of FLT and to meet data retention needs in HPC systems.

Different from the file-centric FLT data retention method, ActiveDR considers the activeness of users at the core of its design. As shown in Figure 2, ActiveDR first evaluates the activeness rank of users by evaluating the operation activeness and the outcome activeness. Then it classifies all users into four classes, i.e. “Both Active” , “Operation Active Only” , “Outcome Active Only” and “Both Inactive.” By scanning the user directories in an ascending order of the user activeness, ActiveDR prioritizes active users over inactive users in retaining their files. In other words, ActiveDR cuts back the file lifetime of inactive users and rewards active users with more file lifetime using the activeness rank. The ActiveDR solution is designed to be automated. Administrators only need an initial setup, then the remaining procedures are automated. Additionally, ActiveDR also supports the commonly needed purge exemption

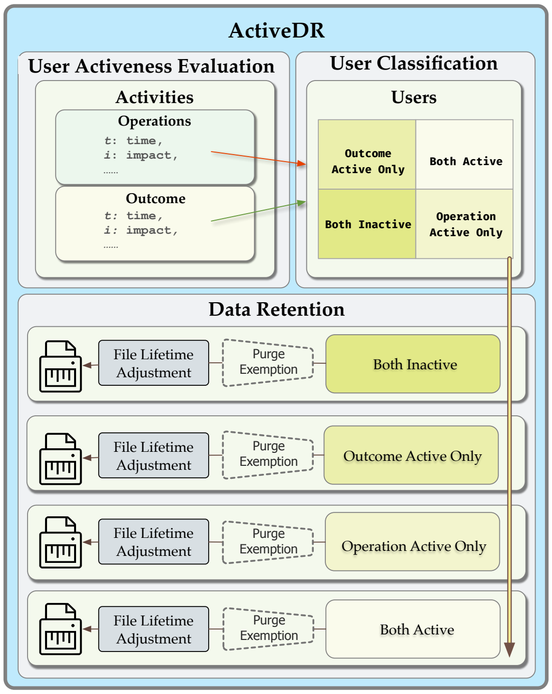


Figure 2: Overview of ActiveDR design. ActiveDR first evaluates the activeness rank of users and classifies them into four classes, i.e. Both Active, Operation Active Only, Outcome Active Only, and Both Inactive. Then it scans user directories in an ascending order of the user activeness and prioritizes active users over inactive users in retaining files, i.e. adjusts the file lifetime of inactive users and rewards active users with more file lifetime using the activeness rank. ActiveDR supports the purge exemption feature.

feature, which allows the administrator to specify a list of files that are requested to retain and skip over these files.

When designing ActiveDR, we consider that system administrators need to focus on daily operations and occasional maintenance and hence should spend minimal effort in tuning system management software. Therefore, in our design, we do not attempt to predict users' future activeness or future file access patterns because users' activity is hard to be predicted precisely, if not impossible. Although there are many studies on predicting user behaviors using machine learning (ML) methods [12, 36, 46], they all require a complicated training process which makes these methods expensive for a rapid evaluation of users' activeness. Additionally, tuning the ML models can be a challenging task for system administrators. Furthermore, the result of many ML approaches is not as intuitively explainable as what system administrators need.

3.1 Activeness-based Perspective

With the goal of capturing the mutual impact between users' activities and file accesses, we consider the data retention problem from a novel activeness-based perspective where users' activities are categorized into two dimensions: *operations* and *outcomes*. Our definition of an *operation* applies to a wide range of user activities performed on the system, as shown in Table 2. These operations reflect the activeness of users and thus change the priorities in the data retention process. Likewise, an *outcome* refers to an accomplishment users have achieved by using the HPC system, or, in other words, what the users produce or generate after performing the operations on the system (examples are also shown in Table 2). The consideration of operation and outcome activities is the hallmark of the activeness-based perspective. It ensures the fairness of user activeness evaluation and prevents the "periodic-file-touch" tricks. Since many HPC facilities do keep track of (or, at least are considering tracking) user operations and outcomes [16, 20, 24, 29, 33, 44], it is a fair assumption and feasible approach to include the consideration of operations and outcomes for the data retention solution.

Table 2: Examples of user activities

Type of User Activity	Examples
Operations	Job submission
	Shell login
	File access
	Data transfer operation
	...
Outcomes	Successful completion of a job
	Successful completion of a task in a workflow
	Dataset generated from a job execution
	Publications resulted from a job output
	...

The advantage of activeness-based perspective is its inclusiveness for a diverse spectrum of users' activities with a particular focus on HPC system and its storage space, which allows flexible choice of user activities for user activeness evaluation. In turn, this approach makes it possible to provide a holistic consideration of different factors including temporal locality, spatial locality, users' behavior and system utilization. For example, by capturing data transfer or data sharing activities among users, the shared use of data files are considered. By capturing file access activities, the temporal locality is considered. By capturing activities that access users' directories, the spatial locality is considered. Next, we discuss the model of activeness evaluation.

3.2 User Activeness Evaluation

To provide a simple and effective solution, the user activeness evaluation algorithm is designed to unify the activeness measurement of different user activities. For any type of activity, the user activeness evaluation algorithm only needs two essential measures which are the time and the impact of the activity. For a specific type of operation, the time and the impact can be concrete metrics. As an example, for a job submission activity (operation), the time can be the job submission time or the job start time. The impact can be the total run time or the CPU hours. Similarly, for an outcome activity such as a publication, the time can be the time of the publication and the impact can be the citation count of the publication. With such a unified activeness measurement model, we are able to quantify and calculate the activeness of users based on their activities. In other words, operations and outcomes can be configured by system administrators based on what they keep track of and with weights to quantitatively measure the impact. Please note that such a setup is only a one-time configuration. ActiveDR uses these information to calculate the activeness to make the data retention decision, which provides an optimized control of purging files instead of solely based on timestamps.

Table 3: List of notations in user activeness evaluation

Description	Notation
Set of n Activity Types	$T = \{\lambda_0, ..., \lambda_{n-1}\}$
Set of k Activities for Activity Type λ	$A_\lambda = \{a_0, ..., a_{k-1}\}$
Activeness of an Activity a_x	D_{a_x}
Set of m Periods	$P = \{p_0, ..., p_{m-1}\}$
Average Activeness of All Activities in A_λ in each period	$Avg(D_{A_\lambda})$
Activeness Ratio of a Certain Period p for Activity Set A_λ	$b_p = D_p / Avg(D_{A_\lambda})$
Period Index of Activity a_x	$e = m - (a_x.ts - a_0.ts)/l + 1$ (ts denotes timestamp)

We now introduce how to calculate the activeness of a user. As shown in Table 3, suppose a user may have n types of activities, we denote the set of all activity types as $T = \{\lambda_0, ..., \lambda_{n-1}\}$. For a certain activity type λ , we consider that there is a set of k activities $A_\lambda = \{a_0, ..., a_{k-1}\}$. For an activity a_x, we consider its activeness to be D_{a_x}. Since the activeness of each activity is measured by its impact, the value of D_{a_x} is a specific, predefined value configured by system administrators. We consider that all activities of type λ are distributed among a set of m periods, and each period contains d days. While the period length d is a configurable parameter, for the activity set $A_\lambda = \{a_0, ..., a_{k-1}\}$ of type λ, the total number of periods m_λ can be calculated as:

$$m_\lambda = \left\lceil \frac{a_{k-1}.ts - a_0.ts}{to\_ts(d)} \right\rceil \tag{1}$$

where function to_ts converts the period length into the same unit as the activity timestamp. We further calculate the average activeness of all k activities of type λ across all m_λ periods:

$$Avg(D_{A_\lambda}) = \frac{\sum_{i=0}^{k-1} D_{a_i}}{m_\lambda} \tag{2}$$

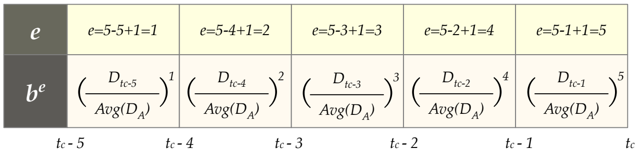
Afterwards, we calculate the activeness of all activities in A_λ during each period p. For a period p_e, let $D_{p_e} = \sum_{i=0}^{j-1} D_{a_i}$ be the overall activeness of all j activities $\{a_i | i \in [0, j)\}$ of type λ occurred in this period, we calculate the activeness ratio of the j activities of type λ in this period:

$$b_{p_e} = \frac{D_{p_e}}{Avg(D_{A_\lambda})} \tag{3}$$

When the activeness ratio $b_{p_e} \geq 1$, we consider that the user is active on type λ activities during period p_e. When the activeness ratio $b_{p_e} < 1$, we consider that the user is inactive on type λ activities during period p_e.

Let t_c be the current time when the user activeness evaluation begins. For an activity a_x occurred during the period e, we calculate the index of the period e by the following equation:

$$e = m_\lambda - \left\lceil \frac{t_c - a_x.ts}{to\_ts(d)} \right\rceil + 1 \tag{4}$$



e	e=5-5+1=1	e=5-4+1=2	e=5-3+1=3	e=5-2+1=4	e=5-1+1=5
b^e	$\left(\frac{D_{tc\text{-}5}}{Avg(D_A)}\right)^1$	$\left(\frac{D_{tc\text{-}4}}{Avg(D_A)}\right)^2$	$\left(\frac{D_{tc\text{-}3}}{Avg(D_A)}\right)^3$	$\left(\frac{D_{tc\text{-}2}}{Avg(D_A)}\right)^4$	$\left(\frac{D_{tc\text{-}1}}{Avg(D_A)}\right)^5$

Figure 3: Time-series activeness rank vector when $m_\lambda = 5$.

Then, a time-series activeness rank vector is built as shown in Figure 3. The length of the vector is equal to the total number of periods that the specified type of activities span over, i.e. m_λ. Each element in the vector represents an activeness rank of the corresponding period. The ActiveDR is designed to value those users who remain active recently. Therefore, at time t_c, we expect the activeness rank acquired from a closer period to have a larger impact against the overall activeness rank. As such, we consider the activeness rank from the eth period to be $(b_{p_e})^e$. The more the period p_e is closer to the current time t_c, the larger the value of e will be, and hence the more the activeness ratio in period p_e contributes to the overall activeness rank. This feature is guaranteed by the monotonic property of exponential function.

Finally, after the activeness rank vector is derived, the overall activeness rank of a particular activity type λ is calculated as:

$$\Phi_\lambda = \prod_{e=1}^{m} (b_{p_e})^e \tag{5}$$

With this equation, ActiveDR guarantees that the activeness rank Φ_λ is either in the range $[0, 1)$ or in the range $[1, +\infty)$. We consider that when $0 \leq \Phi_\lambda < 1$, the user is inactive for the activities of type λ, and when $\Phi_\lambda \geq 1$, the user is active for the activities of type λ. In addition, the larger the value of Φ_λ is, the more active the user is for this type of activity, and vice versa.

In ActiveDR, we consider two classes of user activities, operations and outcomes. For m_{op} types of operation activities and m_{oc} types of outcome activities, we can perform the following calculation to derive the overall operation activeness rank Φ_{op} and the overall outcome activeness rank Φ_{oc}:

$$\Phi_{op} = \prod_{\lambda_{op}=1}^{m_{op}} \Phi_{\lambda_{op}} \quad and \quad \Phi_{oc} = \prod_{\lambda_{oc}=1}^{m_{oc}} \Phi_{\lambda_{oc}} \tag{6}$$

where λ_{op} denotes an operation activity and λ_{oc} denotes an outcome activity. It is noteworthy that both Φ_{op} and Φ_{oc} will be within either the range $[0, 1)$ or the range $[1, +\infty)$ since the activeness of each activity in these categories is within either of these two ranges as well.

Please note that, in ActiveDR design, we have considered the case that outcomes may need longer time to be yielded. That is why the user activeness evaluation model is based on m consecutive periods, instead of just one period (please see Equations (2) - (6) in this section). Additionally, the activeness ratio of each activity type is calculated as an average value in each period, as shown in

Equation (3). Therefore, long jobs would not be penalized because of their long span of run time.

3.3 User Classification

Based on user activeness evaluated, ActiveDR classifies all users into four categories, as depicted in Figure 4. In each category, users are sorted according to their activeness rank. The data retention procedure will scan users' directories based on these four different user activeness categories. In ActiveDR, the operation activeness rank Φ_{op} is given higher priority. Therefore, users are sorted according to their operation activeness first and then are further differentiated according to their outcome activeness.

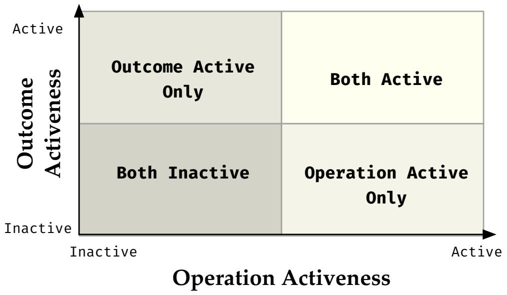


Figure 4: User classification matrix

3.4 Data Retention

To run the data retention procedure, the administrator needs to provide an initial file lifetime for new users and the both-inactive users so that the files of these users will follow the initial file lifetime setting and will not be purged when they are scanned the first time. The administrator also needs to provide a purge target indicating the space utilization that should be reached.

Optionally, the administrator can specify a list of reserved files for the purpose of file purge exemption. ActiveDR reads the file reservation list and stores the paths of the reserved files into a compact prefix tree. When scanning the files of each user, ActiveDR can efficiently determine if the path of an encountered file is in the file reservation list and skip over the reserved files for the retention procedure. Please note that we consider the file reservation list as a contract between users and the system administrator. The paths of the files on the reservation list are not supposed to change. If a user change the file path of a previously reserved file without notifying the system administrator, we consider it means that the user has cancelled the reservation of that file.

Different from the FLT data retention solution where the data files are scanned in the order specified by the system, ActiveDR scans users' directories in an ascending order of the users' activeness rank. First, ActiveDR will evaluate users in both-inactive and outcome-active-only categories. Afterwards, ActiveDR visits the other two categories, i.e. operation-active-only and both-active, in an ascending order of the outcome activeness.

When visiting users in a certain activeness category, ActiveDR scans each file in the user's directory. For each file that is not reserved, ActiveDR adjusts the lifetime of the file by multiplying it with the activeness rank of the user (shown as "file lifetime adjustment" in Figure 2). The more active the user is, the higher chance his/her files will survive from being purged. Suppose the initial file lifetime is d days, the adjusted file lifetime ε_f of file f owned by a user is calculated by the following equation:

$$\varepsilon_f = d \times \Phi_{op} \times \Phi_{oc} \tag{7}$$

At time t_c, ActiveDR examines the access time $atime_f$ of file f and purges the file as long as $t_c - atime_f > \varepsilon_f$.

At any time when the purge target is reached, ActiveDR will stop the data retention procedure. To ensure that active users are protected from file purge to the maximum degree, each time when finishing the file purge scanning of an activeness group, ActiveDR will test if the purge target is reached or not. If not, ActiveDR will retrospectively work on that activeness group for a specified number of times (currently five times in our implementation) and decrease the user activeness rank by a predefined certain percentage each time (currently 20% in our implementation). If the purge target is still not reached after all activeness groups are tried, ActiveDR will stop and report to the administrator via specified reporting mechanism.

In the current design, the lifetime of a file is only extended by its owner's activeness rank even though the file may be shared by other users. We keep such a design because we consider that the owner of the file should be responsible for the files when he/she shares them. We consider that such a design principle can help with suppressing the complexity of the solution.

When ActiveDR is used for a new system or in the case where new users accounts are created in HPC systems, it is highly possible that no activity information is available for all or some users. To avoid the files of these users being purged immediately after the first run of ActiveDR, we set the initial user activeness rank of all activity types to be 1.0. This handling ensures that new users' files are provided with the initial file lifetime and will not be purged during the first data retention process.

4 EVALUATION

In this section, we first introduce our dataset, experimental platform, and our evaluation procedure, then we introduce our experimental results, including the data retention results and the performance of ActiveDR.

4.1 Experimental Setup

4.1.1 Datasets. We used a dataset from Oak Ridge Leadership Computing Facility (OLCF) that contains a series of system traces recorded during 2015 and 2016 from the Titan supercomputer. The system traces include the job scheduler logs collected from 2013 to 2016 (1,368,398 job submissions), application logs from 2015 to 2016 (1,040,886 file paths and 4,973,011 application executions), the list of 13,813 anonymized system users in 2016, the weekly metadata snapshots of the Spider file system from 2015 to 2016 (over 2TB compressed), and the list of 1,151 publications published from 2013 to 2016 by the users at OLCF (the publication list is provided by OLCF too). As the file size is not directly available and we can only get the number of stripes from the metadata snapshot, we generate a synthesized file size for each file in the snapshot according to the best striping practice of the Spider file system suggested by [30].

4.1.2 Experimental Platform. We conducted our emulation-based evaluation on the Cori supercomputer hosted at the National Energy Research Scientific Computing Center (NERSC). Specifically, we used the Haswell compute nodes for our experiments. Each Cori Haswell compute node has two 16-core Intel® Xeon™ processors E5-2698 v3 ("Haswell") at 2.3 GHz and 128 GB of DDR4 2133 MHz memory. The peak performance of each compute node is 1.2 TFlops. The compute nodes use GPFS for its home directory and multiple Lustre file systems as scratch spaces. We used a 30 PB Lustre file system with over 700 GB/s peak I/O bandwidth for our evaluation. Our ActiveDR implementation is written in Python along with the `mpi4py` package to enable parallel emulation. Other python packages we used include `pandas` and `numpy`.

4.1.3 Evaluation Procedure. In our evaluation, we designed an emulation-based experiment to verify the effectiveness and efficiency of ActiveDR, in comparison with the FLT data retention method. Throughout the entire experiment, we set the purge target to be 50% of the total storage capacity (the total synthesized size of all files in the last weekly metadata snapshot of 2015). Also, we used the job scheduler logs as the input for operation activities, and we used the research publication list as the source of outcome activities. In particular, for each job, we use the core hours (number of CPU cores multiplied with the job duration) as the activeness score. Also, we derive the activeness score of each publication by calculating the multiplication of adjusted citation count ϕ and the rank of the user in the author list θ. Given the actual citation count c, the total number of authors of each publication n and the index of the author in the author list i, the activeness of each publication D_{pub} can be calculated as follows:

$$D_{pub} = \phi \times \theta = (c + 1) \times (n - i + 1) \tag{8}$$

Our evaluations were conducted using these two activity traces in hope to show the effectiveness of ActiveDR in the situation where the outcome activities are not directly related to the operation activities, but please note that ActiveDR can work with different types of activities, as discussed in Section 3.1.

To initialize our experiment, we first load the last weekly metadata snapshot in 2015, extract the file paths and index them into a compact prefix tree along with the synthesized file size information generated. The compact prefix tree serves as a virtual file system in our emulation. It allows us to test if a given file path matches with an existing file and also enables us to efficiently retrieve the size information of each file with the corresponding file path. We then replay the application logs of 2016 and emulate the file accesses and data retention processes. We run ActiveDR and FLT solutions with 90-day file lifetime and a 7-day purge trigger interval on the weekly metadata snapshots. These settings were previously used in the Spider II storage system at OLCF, and we reuse such settings to restore the real data retention situation as much as possible.

During the experiments, each time when the ActiveDR data retention is triggered, we first run a preparation procedure to load the corresponding weekly metadata snapshot as well as the activity traces, then evaluate the user activeness and store the corresponding user activeness in memory. Each time when our emulator encounters a file path during the process of replaying the application logs, we first test whether the file is already indexed in the compact prefix tree. If not, we count a file miss; otherwise, we follow the data retention procedure in our design to remove the file. For FLT data retention, we replay the logs and purge the files as in the logs. There is no preparation procedure for FLT.

Since the Spider metadata snapshot has already been a result of the 90-day FLT data retention, we also tested both data retention solution with 7-day, 30-day and 60-day file lifetime, which are shorter than the 90-day file lifetime. This evaluation allows us to observe how both data retention solutions perform with different file lifetime configurations. In addition, we still include the result with the 90-day file lifetime configuration in order to understand what percentage of file misses can be reduced by using ActiveDR as opposed to FLT.

By utilizing the `mpi4py` package, we are able to use multiple processes working together to scan the metadata snapshot. Each process maintains a series of counters to record the number of purged/retained files, the total size of the purged/retained files, and the number of users whose files are purged/retained, etc. Meanwhile, we set multiple probes to monitor the running time and the memory consumption of the program.

4.2 User Activeness

ActiveDR evaluates user activeness before carrying out the data retention procedure. As shown in Figure 5, we can see that the entire user space can be divided into four categories as shown in the activeness matrix. Among 13,813 users, only 0.4%-0.9% users are in the both-active category. The percentage of the operation-active-only users slightly increases from 1.1% to 3.5% as the period length grows from 7 days to 90 days, and the percentage of the outcome-active-only users slightly declines from 3.4% to 2.9%. Most of the users are both inactive (accountable for 92.7% - 95%). For the vast majority of users who are inactive for both operations

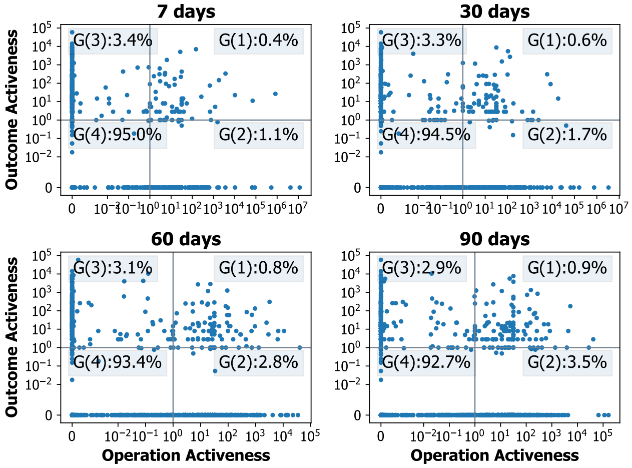


Figure 5: User activeness matrix. Letter "G" in the figure means "activeness group".

and outcomes, their files are considered to be the high-priority candidate for purging. Therefore, when a specific purge target is given, ActiveDR takes advantage of such highly-skewed user distribution among different activeness levels and start the data retention process from purging the files of these inactive users. As compared to FLT, ActiveDR can reach the purge target with more files purged from inactive users. Therefore, more files of active users are expected be retained.

4.3 File Miss Reduction

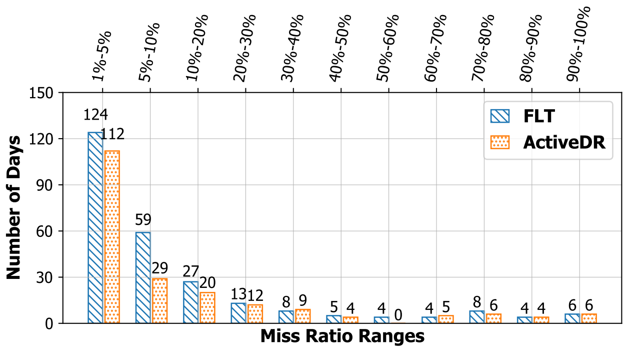


Figure 6: File miss ratio distribution by number of days

Figure 6 shows a comparison between FLT and ActiveDR in terms of the file miss ratio distribution throughout 366 days in 2016. As can be seen from the figure, with ActiveDR, the number of days with 1%-5% file misses is roughly reduced by 10% and the number of days with 5%-10% file misses is almost reduced by half. Overall, the number of days with more than 5% file misses is reduced by 31%, from 138 days to 95 days [§]. This result shows that, on a yearly basis, ActiveDR reduced the total time during which users may randomly suffer from over 5% file misses from half year to only a quarter. Except for the file miss ratio range within 30%-40% and the one within 60%-70%, we see 1 to 4 days of reduction on the file miss ratio ranges over 20%. Additionally, ActiveDR successfully reduced the number of days with 50%-60% file misses from 4 to 0. Considering the fact that re-transmission or re-generation of a file upon a file miss can be very expensive, even such a small reduction can be highly beneficial.

We report the file miss reduction of each user group in Figure 7. Overall, the number of file misses for both FLT and ActiveDR shows an uprising trend with major increases in almost every 3 months of the year. This is because of two reasons. First, when initializing the virtual file system, we only load the last weekly metadata snapshot in 2015 which is already a snapshot of a data retention result produced by the OLCF 90-day retention solution. Thus, in our experiment result, the number of file misses remains small for the first few months of 2016. Second, as the weekly data retention performs with a 90-day file lifetime setting, more and more files are deleted, which leads to an increasing number of file misses. However, while both file miss numbers of FLT and ActiveDR increase with a growing number of data retention operations being performed, we can see that the number of reduced file misses by ActiveDR, i.e., the gap between FLT and ActiveDR, also grows, for all four types of users. In general, the result indicates that ActiveDR helps reduce the file misses in the long run, as compared to FLT solution.

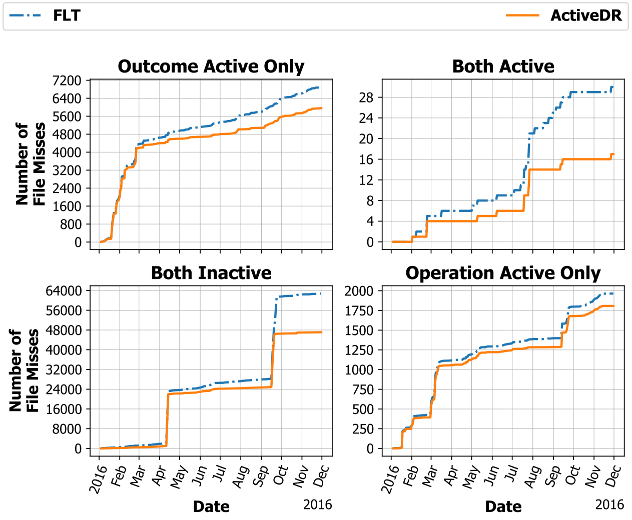


Figure 7: File miss reduction in user activeness matrix

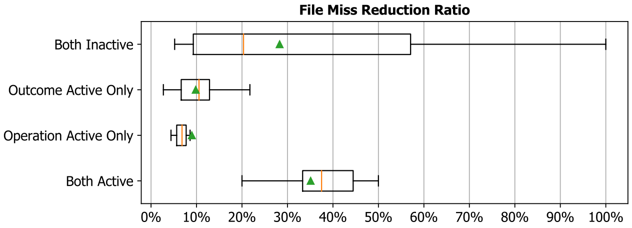


Figure 8: Statistics on file miss reduction ratio

Figure 8 reports the statistics about the file miss reduction ratio, which is the percentage of file miss reduction introduced by replacing FLT with ActiveDR. On average, as indicated by the green triangles in the box plot, ActiveDR is able to reduce 37% of the file misses for both-active users, 7.5% for operation-active-only, 11.2% for outcome-active-only and 27.5% for both-inactive, as compared to the traditional FLT data retention method. While ActiveDR achieves the largest average file miss reduction ratio for both-active users, it surprisingly achieves the second largest one for the group of both-inactive users, with the maximum reduction of 100%, which is twice large as that of both-active category.

When examining the details, we found that the number of FLT file misses for both-inactive users remains very small during the first 4-5 months. However, the number of file misses with the ActiveDR solution is even smaller and sometimes zero in several days

[§] This is calculated by summing up all the number of days in each miss ratio range that is larger than 5%.

during these months. This result leads to the 100% file miss reduction ratio and overall higher miss reduction ratio than those of outcome-active-only and operation-active-only users. Such a phenomenon exactly shows the sensitivity of our approach in reflecting the users' activeness, because the file miss was very low during the very first few months of 2016. In our emulation, we started data retention iterations from the beginning of 2016 and not many recently accessed files were purged at that time.

Additionally, we can observe that ActiveDR acts very similarly to FLT for operation-active-only users, because only operational activities are considered for those users. This observation also shows a positive sign about the relevance between operations and file accesses, since we only use the job scheduler logs as the source of operations and no file access traces were used. In comparison, we can see the benefits of considering outcome for the outcome-active-only users, as this user group experiences noticeably less file misses than operation-active-only users, attributing to ActiveDR.

4.4 Retention with Various Lifetime Settings

We also investigate how ActiveDR and FLT behave differently when the file lifetime is set to 7, 30, 60 and 90 days, respectively. We analyze how much data can be retained by each solution. For this purpose, we select the retention result on the last weekly metadata snapshot we have (which was captured on Aug 23rd of 2016) to examine the details.

ActiveDR prioritizes active users and their file availability. Therefore, when a specific data retention target is given, we expect that ActiveDR should retain more files for active users and less files for inactive users. In our evaluation, we set up the purge target to be 50% of the total capacity. With this purge target, we run both FLT and ActiveDR to observe the retention result for the users of various activeness categories.

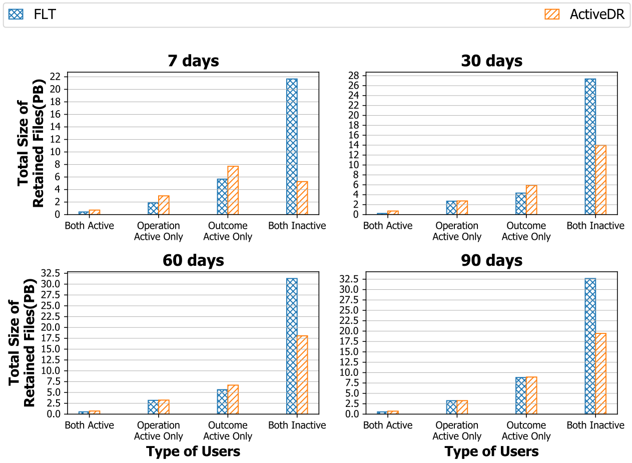


Figure 9: Total size of retained files for users of various activeness categories

As shown in Figure 9 and Table 4, for both-inactive users with any period lengths, ActiveDR retains 13PB - 16PB less data than FLT. The saved space is about half of the 32PB total capacity of the Spider file system. In addition, for each category of users, ActiveDR retains up to 213.47% more data for active users as shown in Table 4 (about 10TB to 2PB more files across different period lengths as shown in Table 5).

Table 4: Percentage of file size that ActiveDR retains more than FLT

Period Length (days)	Both Active	Operation Active Only	Outcome Active Only	Both Inactive
7	71.42%	59.80%	36.07%	-75.67%
30	213.47%	1.66%	35.66%	-48.96%
60	36.32%	1.85%	18.62%	-42.24%
90	33.58%	0.32%	1.33%	-40.48%

Table 5: Difference between the total size of files retained by ActiveDR and FLT (in PB)

Period Length (days)	Both Active	Operation Active Only	Outcome Active Only	Both Inactive
7	0.299	1.120	2.044	-16.390
30	0.490	0.045	1.547	-13.393
60	0.192	0.059	1.051	-13.222
90	0.181	0.011	0.118	-13.223

It is worth noting that the metadata snapshot we use is already a result of the 90-day FLT data retention at OLCF, and a significant number of files were already purged from the file system when the metadata snapshot was captured. Therefore, ActiveDR was only able to evaluate a limited number of files from the remaining files as what should be retained given larger period length settings such as 60 days and 90 days. This explains the declining trend of file retention difference shown in Table 4 and Table 5 as the period length increases. Also, given large period length settings (such as 60-day and 90-day), the retention difference remains relatively insignificant for both-active users and operation-active-only users as compared to that of smaller period length settings. This is because almost every job submission can result in renewing the access time of files that the job accesses, and the file access time is exactly what the 90-day FLT retention solution at OLCF monitors. Therefore, the impact of the job activeness and the file access recency remains similar. Moreover, the retention difference for the outcome-active-only users given larger period length settings is still remarkably larger than that of both-active and operation-active-only users. This is because the number of outcome-active-only users is either larger than that of both-active users or close to the number of operation-active-only users (as shown in Figure 5). Therefore, it is normal that the total size of files retained for the outcome-active-only users is larger than that of the other two types of users. In fact, this result exactly shows the benefits of considering the "outcome" perspective.

We also compare the total size of files purged by ActiveDR and FLT in Figure 10 and Table 6. We can see that, as compared to FLT, ActiveDR purges fewer files for all active users, and purges more files for both-inactive users for 7-day and 30-day period lengths. The file purge effect of ActiveDR remains about the same as FLT for 60-day and 90-day period lengths. Still, as the period length grows, we observe a declining trend in the file purge difference and we attribute this to the fact that the metadata snapshot we

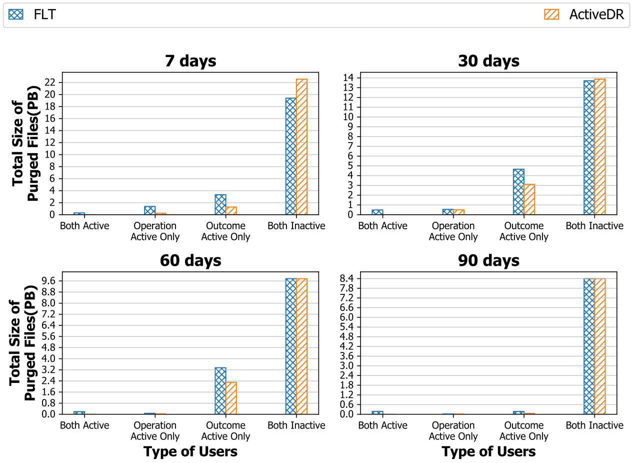


Figure 10: Total size of purged files for the users of various activeness categories

Table 6: Difference between the total size of files purged by FLT and ActiveDR (in PB)

Period Length (days)	Both Active	Operation Active Only	Outcome Active Only	Both Inactive
7	0.299	1.120	2.044	-3.167
30	0.490	0.045	1.547	-0.170
60	0.192	0.059	1.051	0.001
90	0.181	0.010	0.117	0.00007

use is already a result of the 90-day FLT data retention at OLCF. Also, we can see that the file purge differences for all activeness types of users are exactly the same as the file retaining differences shown in Table 5. However, in terms of the purge differences for both-inactive users, the numbers are much smaller. We can see that ActiveDR does not lose the ability to purge files for inactive users and actually performs better than FLT.

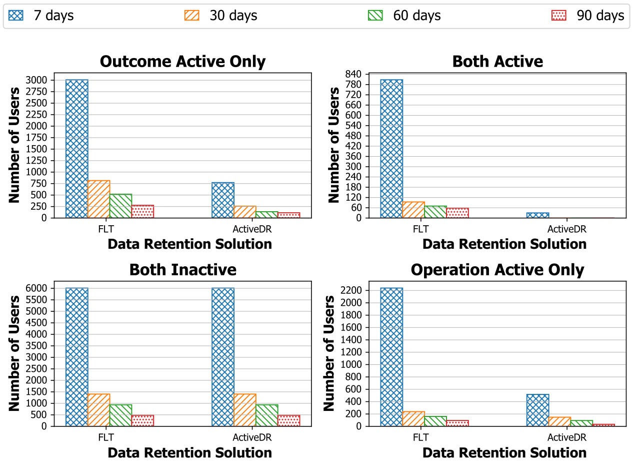


Figure 11: Number of users affected by file purge

As shown in Figure 11, by adopting ActiveDR, the number of users affected by data purge actions in all three active user groups is much smaller than that of the FLT approach. Specifically, the number of both-active users affected by file purge actions is less than 60, while such number for the FLT approach is over 700 when the period length is 7 days. This result shows that ActiveDR can protect active users from data loss caused by file purge operations.

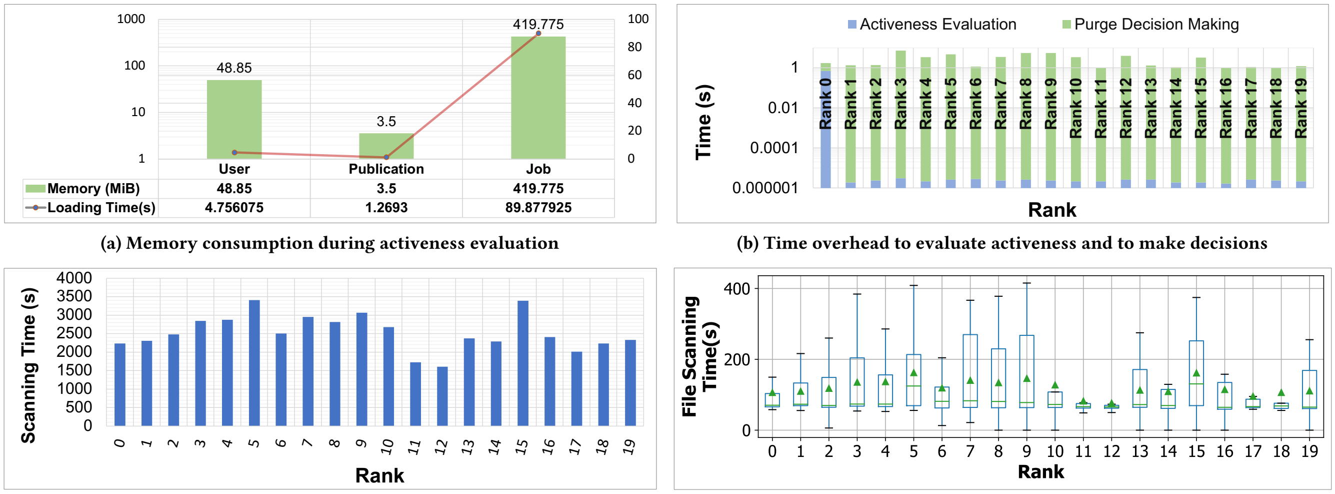
4.5 Performance Evaluation

As a data retention solution aiming to be used in real systems, we expect ActiveDR to be efficient in terms of both time and space complexity. We report the performance evaluation result in Figure 12. From Figure 12a, we can see that ActiveDR only consumed 48.85MB memory for the user list, 3.5 MB for the publication list and 419.77MB for the job traces. The total time for loading these traces is only 1 minute and 35 seconds. Figure 12b shows that, when executing in parallel mode, the main process takes 700 ms for activeness evaluation while other processes only take a few microseconds to perform the activeness evaluation. All processes accumulatively take 1 to 5 seconds for making purge decision for all 1,040,886 files recorded in the application log. Since file access pattern is no longer a necessary consideration in ActiveDR when evaluating user activeness, we can avoid loading gigabytes of metadata snapshots in real practice. Instead, we load the job activity trace, which only accounts for hundreds of megabytes. This further ensures the rapid process for user activeness evaluation and for making purge decision. When testing the purge effect with different period length settings on a single metadata snapshot, it took about 1 hour to scan the entire metadata snapshot with multiple parallel processes, as shown in Figure 12c. As the metadata snapshot is stored as a series of gzipped text files, each process took about 50 to 400 seconds to scan each file, as shown in Figure 12d.

5 DISCUSSION

This research study aims to provide a novel data retention strategy that values the data accessibility of active users and promotes fruitful use of HPC system. In our evaluation, we selected job submissions as operation activities and selected the publications as the outcome activities. We made such a choice for two reasons. First, we hope to select a type of operation activity that users perform in the HPC system but it does not have to be directly relevant to any file properties. Also, we hope to select a type of outcome activity unlike job completion or data generation that can be easily captured inside the HPC system. Rather, we would like to select a type of outcome activity that user perform outside the purview of HPC system. In other words, we would like to show how diverse the user activity types can be, with the objective of being practical still. Second, the dataset we have allows us to explore such an interesting combination of operation activities and outcome activities, but also limits us to explore other types of operation activities or outcome activities such as data transfer or data generation.

However, it is noteworthy that the system administrator can choose any type of operation activity and any type of outcome activity which are appropriate for their own system settings. There is no limitation on the type of activities as long as the activities are trackable with occurrence timestamp and quantifiable impact factor.



(a) Memory consumption during activeness evaluation

(b) Time overhead to evaluate activeness and to make decisions

(c) Total file scanning time of each process

(d) Single file scanning time variations of different processes

Figure 12: Performance evaluation

For operation activities, we suggest choosing the ones that can be easily tracked through various logs and traces. When choosing the types of operations, we suggest considering whether the actual data retention strategy is more relevant to file properties. Likewise, when choosing the types of outcomes, we suggest considering whether the resulting data retention strategy is more sensitive to the completion of activities performed on an HPC system or it is more sensitive to other accomplishments that users get outside the purview of the HPC system.

Once the operation activities and the outcome activities are selected, the system administrator can utilize various techniques to collect the traces about the selected activities. The system administrator can either utilize logs or traces that are readily available in the HPC system or develop scripts or tools to facilitate tracing activities automatically. Also, if the system can tolerate inaccurate user activeness evaluation to some extent, there is no limitation on the application of manually collected activity traces as well. For example, for the sample operation activities and outcome activities listed in Table 2, most of them can be tracked via readily available logs and traces in the HPC system, such as job submission and job completion (via job scheduler logs), file access and dataset generation (via PFS logs and job scheduler logs). Some of them may need efforts in configuring or developing monitoring tools, such as shell login, data transfer, task completion in a workflow. Others, such as publications resulted from job output, may be captured with a combination of automated solutions (e.g., job-related user ID extractor and publication database crawler) along with additional manual efforts (e.g., manual auditing). Please note that the major focus of our study is to propose an activeness-based data retention solution rather than proposing any activity tracing mechanism. We provide the above discussion as a suggestion or a starting point for any system administrator who might be interested in applying our method in practice. The system administrators eventually have the due right to choose the most appropriate activity types and the corresponding activity tracing methods that meet the need of their system accordingly.

ActiveDR promotes fruitful use of HPC storage space. We currently consider it is a good practice if the users just access their files according to their inherent needs. For most cases, we suggest that the users should actively perform operations and/or generate outcomes and naturally benefit from the convenience ActiveDR provides. However, if the users need to be aware of the file lifetime settings and need to plan for backing up important data files, we suggest that the system administrators can provide the purge trigger interval to the users as a reference.

In our evaluation, we ran our prototype implementation as a regular job. But our prototype implementation proved that our method can be implemented as a parallel program working on HPC systems. In actual practice, the system administrator can implement their own version, which adapts to their data retention workflow and handles the fault-tolerance issue according to their system specifics.

ActiveDR is not only unique as compared to state-of-the-art data retention strategies. The superiority of ActiveDR lies in its consideration of user activities, its low cost of implementation, and its practicality.

6 CONCLUSION

Existing data retention methodologies on HPC systems either are limited by compromised efficacy or ignore the dynamics of users' activities and hence undermine the file availability to users. In this study, we rethink the data retention problem from the activeness-based perspective which holistically captures users' activeness. We have introduced ActiveDR, an activeness-based data retention solution which is unique, effective, and reproducible with the following characteristics: 1) user-friendly: in ActiveDR, we consider users and their activities at the core of its design. The activeness-based perspective holistically captures both operations that users perform on the system and the outcomes that users yield by using the system.

We value the user experience of the scratch space and we aim to reduce the file misses for active users; 2) administrator-friendly: the definition of operations and outcomes in the activeness-based perspective covers a wide range of user activities. Therefore, the administrators can simply utilize traces available or activities captured by monitoring tools they have been using to serve the user activeness evaluation. They can customize the ActiveDR as needed too; 3) resource-friendly: ActiveDR provides an efficient activeness evaluation algorithm that only requires some important properties of user activities. As such, the activeness evaluation process of ActiveDR runs very fast and the memory footprint is negligible; 4) HPC-ecosystem-friendly: ActiveDR is the first data retention solution that promotes the active and fruitful use of the HPC system, which helps promote productive use of HPC facilities.

Although our evaluation was performed based on user job submission and publication traces, system administrators can select other appropriate activities for user activeness evaluation. ActiveDR is designed for HPC storage system, but its reproducibility and resource-efficiency make it a valuable reference to meet the data management need of other shared storage systems as well. More importantly, our study offers new insights about HPC storage management problem and can have an impact on new practices in the HPC community.

ACKNOWLEDGMENTS

We are thankful to the anonymous reviewers for their valuable feedback. This research is supported in part by the National Science Foundation under grant CCF-1718336, OAC-1835892 and CNS-1817094. This manuscript has been authored by an author at Lawrence Berkeley National Laboratory under Contract No. DE-AC02-05CH11231 with the U.S. Department of Energy, and has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy (DOE). The U.S. Government retains, and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. Government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

REFERENCES

[1] MG Aartsen, K Abraham, M Ackermann, J Adams, JA Aguilar, M Ahlers, M Ahrens, D Altmann, K Andeen, T Anderson, et al. 2016. Search for Sources of High-Energy Neutrons with Four Years of Data from the Icetop Detector. *The Astrophysical Journal* 830, 2 (2016), 129.

[2] MG Aartsen, M Ackermann, J Adams, JA Aguilar, Markus Ahlers, M Ahrens, I Al Samarai, D Altmann, K Andeen, T Anderson, et al. 2017. Constraints on Galactic Neutrino Emission with Seven Years of IceCube Data. *The Astrophysical Journal* 849, 1 (2017), 67.

[3] Judie Attard and Rob Brennan. 2018. Challenges in Value-Driven Data Governance. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part II.* 546–554. https://doi.org/10.1007/978-3-030-02671-4_33

[4] Judie Attard and Rob Brennan. 2018. DaVe: A Semantic Data Value Vocabulary to Enable Data Value Characterisation. In *Enterprise Information Systems - 20th International Conference, ICEIS 2018, Funchal, Madeira, Portugal, March 21-24, 2018, Revised Selected Papers (Lecture Notes in Business Information Processing, Vol. 363)*, Slimane Hammoudi, Michal Smialek, Olivier Camp, and Joaquim Filipe (Eds.). Springer, 239–261. https://doi.org/10.1007/978-3-030-26169-6_12

[5] Judie Attard and Rob Brennan. 2018. A Semantic Data Value Vocabulary Supporting Data Value Assessment and Measurement Integration. In *Proceedings of the 20th International Conference on Enterprise Information Systems, ICEIS 2018, Funchal, Madeira, Portugal, March 21-24, 2018, Volume 2*, Slimane Hammoudi, Michal Smialek, Olivier Camp, and Joaquim Filipe (Eds.). SciTePress, 133–144. https://doi.org/10.5220/0006777701330144

[6] Ranjita Bhagwan, Fred Douglis, Kirsten Hildrum, Jeffrey O. Kephart, and William E. Walsh. 2005. Time-Varying Management of Data Storage. In *Proceedings of the First Conference on Hot Topics in System Dependability* (Yokohama, Japan) *(HotDep'05)*. USENIX Association, USA, 14.

[7] Buddy Bland. 2014. Present and Future Leadership Computers at OLCF. In *OLCF User Group Conference Call December*, Vol. 3.

[8] Rob Brennan, Judie Attard, and Markus Helfert. 2018. Management of Data Value Chains, a Value Monitoring Capability Maturity Model. In *Proceedings of the 20th International Conference on Enterprise Information Systems, ICEIS 2018, Funchal, Madeira, Portugal, March 21-24, 2018, Volume 2*, Slimane Hammoudi, Michal Smialek, Olivier Camp, and Joaquim Filipe (Eds.). SciTePress, 573–584. https://doi.org/10.5220/0006684805730584

[9] Chi Chen, Zhi Deng, Richard Tran, Hanmei Tang, Iek-Heng Chu, and Shyue Ping Ong. 2017. Accurate Force Field for Molybdenum by Machine Learning Large Materials Data. *Physical Review Materials* 1, 4 (2017), 043603.

[10] Ying Chen. 2005. Information Valuation for Information Lifecycle Management. In *Proceedings of the Second International Conference on Automatic Computing (ICAC '05)*. IEEE Computer Society, USA, 135–146. https://doi.org/10.1109/ICAC.2005.35

[11] Peng Cheng, Yutong Lu, Yunfei Du, and Zhiguang Chen. 2018. Accelerating Scientific Workflows with Tiered Data Management System. In *20th IEEE International Conference on High Performance Computing and Communications; 16th IEEE International Conference on Smart City; 4th IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2018, Exeter, United Kingdom, June 28-30, 2018.* IEEE, 75–82. https://doi.org/10.1109/HPCC/SmartCity/DSS.2018.00042

[12] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems.* 191–198.

[13] Tina M Declerck et al. 2014. Using Robinhood to Purge Data from Lustre File Systems. *Proceedings of the 2014 Cray User Group, Lugano* (2014).

[14] Jeffrey J Donatelli, James A Sethian, and Peter H Zwart. 2017. Reconstruction from Limited Single-Particle Diffraction Data via Simultaneous Determination of State, Orientation, Intensity, and Phase. *Proceedings of the National Academy of Sciences* 114, 28 (2017), 7222–7227.

[15] Matt Ezell, Rick Mohr, John Wynkoop, and Ryan Braby. 2012. Lustre at Petascale: Experiences in Troubleshooting and Upgrading. In *2012 Cray User Group Meeting.*

[16] RIKEN Center for Computational Science. 2021. Expected Outcome: Priority Issues and Exploratory Challenges. https://www.r-ccs.riken.jp/en/fugaku/outcome.

[17] Timothy J. Gibson. 1999. An Improved Long-Term File Usage Prediction Algorithm. In *25th International Computer Measurement Group Conference, December 5-10, 1999, Reno, Nevada, USA, Proceedings.* Computer Measurement Group, 639–648. http://www.cmg.org/?s2member_file_download=/proceedings/1999/9527.pdf

[18] Min-Woo Kwon, JunWeon Yoon, TaeYoung Hong, and ChanYeol Park. 2017. Accelerated Purge Processes of Parallel File System on HPC by Using MPI Programming. In *Advances in Computer Science and Ubiquitous Computing - CSA/CUTE 2017, Taichung, Taiwan, 18-20 December (Lecture Notes in Electrical Engineering, Vol. 474)*, James J. Park, Vincenzo Loia, Gangman Yi, and Yunsick Sung (Eds.). Springer, 1134–1140. https://doi.org/10.1007/978-981-10-7605-3_181

[19] J. Li, S. Singhal, R. Swaminathan, and A. H. Karp. 2012. Managing Data Retention Policies at Scale. *IEEE Transactions on Network and Service Management* 9, 4 (December 2012), 393–406. https://doi.org/10.1109/TNSM.2012.101612.110203

[20] Seung-Hwan Lim, Hyogi Sim, Raghul Gunasekaran, and Sudharshan S Vazhkudai. 2017. Scientific user behavior and data-sharing trends in a petascale file system. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* 1–12.

[21] J. Liu, D. Bard, Q. Koziol, S. Bailey, and Prabhat. 2017. Searching for Millions of Objects in the BOSS Spectroscopic Survey Data with H5Boss. In *2017 New York Scientific Data Summit (NYSDS).* 1–9. https://doi.org/10.1109/NYSDS.2017.8085044

[22] Yaning Liu, George Shu Heng Pau, and Stefan Finsterle. 2017. Implicit Sampling Combined with Reduced Order Modeling for the Inversion of Vadose zone Hydrological Data. *Computers & Geosciences* (2017).

[23] Arun Mannodi-Kanakkithodi, Tran Doan Huan, and Rampi Ramprasad. 2017. Mining Materials Design Rules from Data: The Example of Polymer Dielectrics. *Chemistry of Materials* 29, 21 (2017), 9001–9010.

[24] Marta Mattoso, Jonas Dias, Kary ACS Ocana, Eduardo Ogasawara, Flavio Costa, Felipe Horta, Vitor Silva, and Daniel De Oliveira. 2015. Dynamic steering of HPC scientific workflows: A survey. *Future Generation Computer Systems* 46 (2015), 100–113.

[25] Michael Mesnier, Eno Thereska, Gregory R Ganger, Daniel Ellard, and Margo Seltzer. 2004. File Classification in Self-* Storage Systems. In *International Conference on Autonomic Computing, 2004. Proceedings.* IEEE, 44–51.

[26] Henry M. Monti, Ali R. Butt, and Sudharshan S. Vazhkudai. 2009. /Scratch as a Cache: Rethinking HPC Center Scratch Storage. In *Proceedings of the 23rd International Conference on Supercomputing* (Yorktown Heights, NY, USA) *(ICS '09)*. Association for Computing Machinery, New York, NY, USA, 350–359. https://doi.org/10.1145/1542275.1542325

[27] National Center for Atmospheric Research. 2020. GLADE File Space. https://www2.cisl.ucar.edu/resources/storage-and-file-systems/glade-file-spaces.

[28] National Energy Research Scientific Computing Center. 2020. NERSC Data Management Policy. https://docs.nersc.gov/data/policy/.

[29] National Energy Research Scientific Computing Center (NERSC). 2021. Publications Resulting from the Use of NERSC Resources. https://www.nersc.gov/news-publications/publications-reports/nersc-user-publications/.

[30] Oak Ridge Leadership Computing Facility. 2016. Best Practice @ OLCF. https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Best-Practices-v6.pdf. , 60 pages.

[31] Oak Ridge Leadership Computing Facility. 2020. OLCF Policy Guides - Data Management Policy. https://docs.olcf.ornl.gov/accounts/olcf_policy_guide.html#data-management-policy.

[32] Oak Ridge Leadership Computing Facility. 2020. Storage Overview. https://docs.olcf.ornl.gov/data/storage_overview.html.

[33] Oak Ridge Natinal Laboratory Leadership Computing Facility (OLCF). 2021. OLCF Project Search. https://www.olcf.ornl.gov/leadership-science/project-search/.

[34] Sarp Oral, James Simmons, Jason Hill, Dustin Leverman, Feiyi Wang, Matt Ezell, Ross Miller, Douglas Fuller, Raghul Gunasekaran, Youngjae Kim, et al. 2014. Best Practices and Lessons Learned from Deploying and Operating Large-scale Data-centric Parallel File Systems. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 217–228.

[35] David Paez-Espino, I Chen, A Min, Krishna Palaniappan, Anna Ratner, Ken Chu, Ernest Szeto, Manoj Pillay, Jinghua Huang, Victor M Markowitz, et al. 2017. IMG/VR: A Database of Cultured and Uncultured DNA Viruses and Retroviruses. *Nucleic acids research* 45, D1 (2017), D457–D465.

[36] Kumar Attangudi Perichiappan Perichappan. 2018. Greedy Algorithm Based Deep Learning Strategy for User Behavior Prediction and Decision Making Support. *Journal of Computer and Communications* 6, 6 (2018), 45–53.

[37] Gauri Shah, Kaladhar Voruganti, Piyush Shivam, and Maria Alvarez. 2006. Ace: Classification for Information Lifecycle Management. *NASA Mass Storage Systems and Technologies* (2006).

[38] Arie Shoshani, Alexander Sim, and Junmin Gu. 2004. Storage Resource Managers. In *Grid Resource Management*. Springer, 321–340.

[39] Stephen Strange. 1992. *Analysis of Long-Term UNIX File Access Patterns for Application to Automatic File Migration Strategies*. Technical Report UCB/CSD-92-700. EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1992/6258.html

[40] Texas Advanced Computing Center. 2020. TACC Usage Policy. https://portal.tacc.utexas.edu/tacc-usage-policy.

[41] Lars Turczyk, Marcel Groepl, Nicolas Liebau, and Ralf Steinmetz. 2007. A Method for File Valuation in Information Lifecycle Management. *AMCIS 2007 Proceedings* (2007), 38.

[42] Lars Arne Turczyk, Oliver Heckmann, Rainer Berbner, and Ralf Steinmetz. 2006. A Formal Approach to Information Lifecycle Management. In *Proceedings of 17th Annual IRMA International Conference, Washington DC*.

[43] Lars Arne Turczyk, Oliver Heckmann, and Ralf Steinmetz. 2007. File Valuation in Information Lifecycle Management. In *Proceedings of the Thirteenth Americas Conference on Information Systems, Keystone, Colorado*.

[44] Sudharshan S Vazhkudai, John Harney, Raghul Gunasekaran, Dale Stansberry, Seung-Hwan Lim, Tom Barron, Andrew Nash, and Arvind Ramanathan. 2016. Constellation: A science graph network for scalable data and knowledge discovery in extreme-scale scientific collaborations. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 3052–3061.

[45] Akshat Verma, David Pease, Upendra Sharma, Marc Kaplan, Jim Rubas, Rohit Jain, Murthy Devarakonda, and Mandis Beigi. 2005. An Architecture for Lifecycle Management in Very Large File Systems. In *Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST '05)*. IEEE Computer Society, USA, 160–168. https://doi.org/10.1109/MSST.2005.4

[46] Armando Vieira. 2015. Predicting online user behaviour using deep learning algorithms. *arXiv preprint arXiv:1511.06247* (2015).

[47] Brent Welch and Garth A Gibson. 2004. Managing Scalability in Object Storage Systems for HPC Linux Clusters. In *MSST*. Citeseer, 433–445.

[48] Fons Wijnhoven, Chintan Amrit, and Pim Dietz. 2014. Value-Based File Retention: File Attributes as File Value and Information Waste Indicators. *J. Data and Information Quality* 4, 4, Article 15 (May 2014), 17 pages. https://doi.org/10.1145/2567656

[49] Erez Zadok, Jeffrey Osborn, Ariye Shater, Charles P Wright, Kiran-Kumar Muniswamy-Reddy, and Jason Nieh. 2004. Reducing Storage Management Costs via Informed User-Based Policies.. In *MSST*. 193–197.

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

We conducted our simulation-based evaluation on the Cori supercomputer hosted at the National Energy Research Scientific Computing Center (NERSC). Specifically, we used the Haswell computing nodes for our experiments. Each Cori Haswell compute node has two 16-core Intel®Xeon™ processors E5-2698 v3 ("Haswell") at 2.3 GHz and 128 GB of DDR4 2133 MHz memory. The peak performance of each compute node is at 1.2 TFlops/node. The compute nodes use GPFS for its home directory and multiple Lustre file systems as scratch spaces. We used a 30 PB Lustre file system with over 700 GB/s peak I/O bandwidth for our evaluation. The simulation program is written in Python along with mpi4py package to enable parallel simulation. Other python packages we used in the simulation includes pandas and numpy.

Author-Created or Modified Artifacts:

```
Persistent ID: 10.5281/zenodo.5168853
Artifact name: ActiveDR v1.0.6

Persistent ID: 10.5281/zenodo.5152773
Artifact name: data\_min.tar.gz
```

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: Cori, CPU(2*16-core Intel Xeon processors E52698), Lustre File System

Operating systems and versions: CLE7.0UP00

Compilers and versions: py_compiler (Python 3.8.3)

Applications and versions: ActiveDR v1.0.6 (written in python)

Libraries and versions: mpi4py, numpy, pandas

URL to output from scripts that gathers execution environment information.

```
https://raw.githubusercontent.com/zhangwei217245/Act⌋
↪  iveDR/master/haswell\_env.txt
```