Adaptive Replication Management in HDFS based on Supervised Learning

Dinh-Mao Bui, Shujaat Hussain, Eui-Nam Huh, Sungyoung Lee

Abstract—The number of applications based on Apache Hadoop is dramatically increasing due to the robustness and dynamic features of this system. At the heart of Apache Hadoop, the Hadoop Distributed File System (HDFS) provides the reliability and high availability for computation by applying a static replication by default. However, because of the characteristics of parallel operations on the application layer, the access rate for each data file in HDFS is completely different. Consequently, maintaining the same replication mechanism for every data file leads to detrimental effects on the performance. By rigorously considering the drawbacks of the HDFS replication, this paper proposes an approach to dynamically replicate the data file based on the predictive analysis. With the help of probability theory, the utilization of each data file can be predicted to create a corresponding replication strategy. Eventually, the popular files can be subsequently replicated according to their own access potentials. For the remaining low potential files, an erasure code is applied to maintain the reliability. Hence, our approach simultaneously improves the availability while keeping the reliability in comparison to the default scheme. Furthermore, the complexity reduction is applied to enhance the effectiveness of the prediction when dealing with Big Data.

Index Terms—Replication, HDFS, Proactive Prediction, Optimization, Bayesian Learning, Gaussian Process.

1 Introduction

The evolution of big data has created a phenomenon in application and solution development to extract, process and store useful information as it emerges to deal with new challenges. In this area, Apache Hadoop is one of the most renowned parallel frameworks. Not only is it used to achieve high availability, Apache Hadoop is also designed to detect and handle the failures as well as maintain the data consistency. Coming along with the development of Apache Hadoop, the Hadoop Distributed File System (HDFS) has been introduced to provide the reliability and high-throughput access for data-centric applications. Gradually, HDFS has become a suitable storage framework for parallel and distributed computing, especially for MapReduce engine, which was originally developed by Google to cope with the indexing problems on big data.

To improve the reliability, HDFS is initially equipped with a mechanism that uniformly replicates three copies of every data file. This strategy is to maintain the requirements of fault tolerance. Reasonably, keeping at least three copies makes the data more reliable and more robust when tolerating the failures. However, this default replication strategy still remains a critical drawback with regards to the performance aspect. Intuitively, the purpose of inventing Apache Hadoop was to achieve better performance in data manipulation and processing [1]. Therefore, this purpose should be carefully studied at every component. In the performance perspective, based on the well-known research of delay scheduling [2], if the task is placed closer to the required data source, the system can achieves faster computation and better availability. The metric measures the distance between the task and the corresponding data source can be referred to as the data locality metric. The main reason for the improvement is twofold. First, the network overhead can be reduced on runtime due to the availability of the local data, and so no inter-communication is needed to transfer the required data from the remote nodes. Second, it is clear that the computation can start immediately on the input data which is locally available, and so no extra task-scheduling effort is consumed. Consequently, it is meaningful to say that improving the data locality would immensely enhance the system performance in terms of availability and calculation time.

Although there are some studies on this subject matter, very few proactive solutions are proposed that rigorously consider the nature of the job workload. Due to the fact that workload in Apache Hadoop consists of short and long tasks together, these tasks should be handled fairly to accelerate the computation. Typically, the Fair scheduler and delay scheduling algorithm [2] provide the optimal data locality when the system is filled with head-of-line jobs and short tasks. However, the long tasks, if they are present would not be treated appropriately and thus make the system imbalanced. One solution is to pro-actively prepare the potential replications before scheduling the tasks in order to redirect and balance the computation. To do that, we aim to improve the data locality metric by changing the replication scheme adaptively with regards to the popularity of the data file. Not only is the nature of the access rate taken into account, but the replica placement is also carefully considered. Note that the access rate is defined as the number of accesses in a given unit of time. Subsequently, the data files in HDFS are replicated based on their own access potential as well as the overall status of the system. By definition, the access

D. Bui is with the Computer Engineering Department, Kyung Hee University, Suwon 446-701, Korea (e-mail: mao.bui@khu.ac.kr).
S. Hussain is with the Computer Engineering Department, Kyung Hee University, Suwon 446-701, Korea (e-mail: shujaat.hussain@oslab.khu.ac.kr).
E. Huh is with the Computer Engineering Department, Kyung Hee University, Suwon 446-701, Korea (e-mail: johnhuh@khu.ac.kr).
S. Lee is with the Computer Engineering Department, Kyung Hee University, Suwon 446-701, Korea (e-mail: sylee@oslab.khu.ac.kr).

potential is recognized as how frequently the specific file might be read in the next time epoch. For example, say a file has an access potential of 32 within the period of 5 seconds: this means that the file might be accessed 32 times in the next 5 seconds. Additionally, the anticipated results and access patterns are cached in the knowledge base in order to instantly match and quickly fire the suitable action without having to re-calculate a similar input. Occasionally, each data file can be efficiently replicated by a different but appropriate strategy. Further, in order to maintain the fault tolerance for less frequently accessed data files, an open source erasure code [3] is modified and applied to protect the system from the effects of failures. Finally, by implementing this framework, the task execution time and storage cost can be improved benefiting the productivity of big data systems.

In summary, the main contributions of this research are as follows.

- We designed an adaptive replication management (ARM) system to provide high availability for the data in HDFS *via* enhancing the data locality metric. As a result, the highly local available data improves the performance of the Hadoop system. It is worth noting that the erasure code is applied to maintain the reliability.
- We proposed a complexity reduction method for the prediction technique in both hyper-parameter learning and training phases. This proposed method significantly increases the performance in terms of reaction rate for the replication strategy while still keeping the accuracy of the prediction.
- We implemented ARM in HDFS and did an evaluation in order to practically verify the effectiveness of the proposed method as compared with the state of the art method.

The remainder of this paper is organized as follows. In Section 2, related works relevant to the topic are provided. Section 3 discusses the methodology in detail. Section 4 presents an architectural overview of the proposed approach while Section 5 includes the detail of the prediction mechanism. In Section 6, the performance evaluation is discussed. Conclusions and future work are summarized in Section 7.

2 Related Works

In the replication area, there are two main methods: the proactive approach and the reactive one. For the proactive approach, the Scarlett solution [4] implements the probability as an observation and then calculates the replication scheme for each data file. The storage budget-limitation is also considered as a factor when distributing the replicas. Although this solution follows a proactive approach instead of using thresholds, the access rate of the data file as well as the suitable placement for replicas is not discussed thoroughly.

Likewise in OPTIMIS [5], an interesting solution for anticipating the data file status has been proposed. In this approach, the data file is classified and engaged in the limited replication scenarios based on the algorithmic prediction of the demand for data file utilization. However, the Fourier series analysis algorithm [6], which is usually used in the field of 'signal processing', is chosen for prediction without a compelling proof of the efficacy. As a consequence, this inappropriate choice may result in poor prediction.

For the reactive approach, the cost-effective dynamic replication management (CDRM) method [7] is a cost-effective framework for replication in a cloud storage system. When the workload changes, CDRM calculates the popularity of the data file and determines the location in the cloud environment. However, this technique follows a reactive model. As a result, by using threshold values, CDRM cannot adapt well to the rapid evolution of large-scale systems.

Similarly, DARE [8] is another reactive model of replication for HDFS. In this model, the authors declare that the probabilistic sampling and competitive aging algorithms are used independently on each node to choose a replication scheme for each data file, as well as to decide the suitable location for each replica. However, there are two issues in this approach. First, the problem of long tasks, which exists in a realistic system, is not considered carefully. In fact, the existence of this issue makes the system unstable. Second, the placement of the replication is judged without considering the file access pattern and system capacity of the destination nodes. For these reasons, DARE might not provide the expected effectiveness on some systems.

The elastic replication management system (ERMS) [9] takes into account an active/standby model for data storage in the HDFS cluster by implementing the complex event processing method to classify the data types. The advantage of ERMS as compared with CDRM and DARE is that it dynamically changes the thresholds for metrics based on the status of the HDFS cluster system. In addition, ERMS is equipped with the erasure code to determine and erase unpopular replicas so as to save the storage.

Nevertheless, although CDRM, DARE and ERMS are developed in different ways, all of them encounter the same problems and limitations. Concretely, these solutions try to classify and implement various replicating scenarios for each type of data files by extracting and processing the obsolete information. For that reason, these approaches cannot generate an optimal replication strategy for parallel systems. The detail of this claim is that when some actions are chosen to handle the 'hot' data files, due to high latency and delay, these files may not be 'hot' anymore by the time the actions are engaged. As a consequence, the replicating decision cannot reflect the trends of the data utilization. Additionally, in the ERMS approach, the erasure code configuration is not clearly specified. For that reason, the storage-reliability of this approach is still not verified.

Discussions on erasure code are interesting. Commonly, it is accepted that not only the performance, but also the reliability is the mandatory aspect of HDFS. To fulfill this requirement, the replication and the erasure code are two types of fault tolerance techniques trying to obtain the same goal. While the replication is suitable for enhancing read operation, it suffers from a large storage overhead of up to 200% [10]. Even with the rapid decline in the cost for the storage facility, this overhead has still become the major problem; this is because the volume and velocity of Big Data dramatically increase at a rate faster than the infrastructure, and are required not only for the storage resources but also for the computation and network utilization [11]. As a result, many corporations including Facebook, Microsoft and Google think of the erasure coding approach as an alternative technique for saving the repository storage space while keeping the same level of reliability. In addition, the erasure coding approach has a long history of development in peer-to-peer systems to ensure the optimal fault tolerance with a low cost of storage [12]. The latest result of erasure coding is the applications to the Microsoft Azure Storage [13] as well as the new version of the Google File System along with some modules of HDFS.

Recently, Facebook has started using an open source erasure code, namely HDFS-RAID [14]. The HDFS-RAID uses the famous Reed-Solomon (RS) code, which is derived from the maximum distance separable (MDS) method [15], in order to ensure the reliability. The configuration for this method is RS (10,4), which stands for a group of 10 stripes and 4 parity blocks for each data file. By using this configuration, HDFS-RAID can survive during 4 block failures and creates only 40% overhead for the storage. In terms of system perspective, this method can be seen as robust and storage efficient in comparison to many other replication approaches.

Other methods employed in erasure coding are Pyramid codes [16] and HDFS-Xorbas [10], which follow the approaches of the local reconstruction code (LRC) [17] and the locally repairable codes (LRCs) [10], respectively. Unlike the MDS coding family, Pyramid codes (which are applied in Microsoft Azure Storage [13]) and HDFS-Xorbas offer better repairable features in terms of network bandwidth and disk I/O. However, the disadvantage of these methods is the higher magnitude of storage overhead as compared with HDFS-RAID.

We discuss one more erasure coding solution, the storageCore [3], which is based on HDFS-RAID. In this solution, the authors combine the standard erasure code and RAID-4-like parity to establish the redundancy. By applying this combination, the solution offers an impressive improvement in repairability and read performance. Additionally, the authors also provide the complete source code with specification and simulation as an open source project. This decision makes storageCore suitable for ready integration with other methods in HDFS. However, the disadvantage of storageCore is an increase in storage overhead of up to 20% in comparison to HDFS-RAID.

According to a number of studies on erasure code, it is clear to see that this branch of fault tolerance possesses an important role in maintaining the reliability for Big Data system. However, by containing only one copy of each data block, the erasure coding approaches have to reunify the data blocks remotely. Even when HDFS reads the erasure coded blocks in parallel, the processing time is still lengthened by the unavailable computing node. To solve this issue, most of the approaches choose to utilize the degraded reading, which actually mitigates the unavoidable drawback. However, this point of design actually reduces the throughput and indirectly increases the computation time.

By examining the related works, we have come to the conclusion that although the research on replication and erasure code exists, not many researchers have thoroughly attempted to balance the data locality and the reliability within a reasonable cost of storage resource. Furthermore, since Hadoop is gradually a complex ecosystem, a faster and more adaptive replication mechanism must be developed.

3 Methodology

3.1 Motivation

In many parallel and distributed systems equipped with MapReduce engine, the processing jobs usually comprise a series of consecutive phases, namely map, shuffle and reduce. In the beginning, map phase reads the input from disk and prepares the intermediate data for other phases. Unless the system includes an expensively infinite band of network capacity, which only exists on very large scale computing, the bottleneck between computing nodes is unavoidable. Due to this fact, it would be optimal if the system can colocate map tasks along with the desired data, especially when the data size is large. Unfortunately, MapReduce scheduler is unable to always satisfy this requirement. Replicating uniformly or increasing the replication factor is not the key to accelerate the computation as well as reduce the slot contention and hot-spot issue. Note that the slot contention happens when the number of concurrent tasks accessing the data file surpasses the number of replicas. Consequently, the tasks with no locally available data have to request for remote access or wait for the next available turns on the same data. Obviously, this issue dramatically decreases the system performance. In the other hand, the hot-spot issue, which is recognized as the attractive nodes to many tasks, makes the system imbalanced and wastes the idle computational capability. These issues must be solved to fulfill the capability of big data system, especially.

To minimize the effect of slot contention and hot-spot issue, many approaches choose to improve the data locality and conduct the load balancing as seen in the Related Works section. Nevertheless, as mentioned above, most of these methods are either maladaptive or inaccurate to provide the suitable replication strategies coping with various data access patterns. It is worth noting that beside the growth in storage cost, the diversity of data access patterns is more critical affecting the performance, the replica management and the balance of the system. However, the characteristic of the data access is inadequately studied in the previous works. Thus, this reason motivates us to design a predictive approach (ARM) to truly enhance the data locality with regard to the system utilization and reliability. By proposing ARM, we expect that our study can be useful to any organizations or companies, which are interested in optimizing the performance within an affordable cost.

3.2 Approach analysis

As discussed in the Related Works section, the high data locality is critical to the performance and the availability of HDFS. Theoretically, our prediction technique intends to improve the data locality by creating the individual replication scheme for each data file based on its own access potential. Naturally, some popular data files may have more replicas than others because these files possess more potential to be utilized by various tasks. The percentage of high potential files can be measured in less than 10% [18]. On the other hand, over 90% of data files might have a minimum access potential [18]. As a consequence, the replication for this large percentage of data files should be limited, as they are clearly not necessary to perform the prediction. A suitable strategy in this situation is to leave these low access potential files with only one replica. This strategy might save some space as well as reduce the computation, but it also reduces the reliability and puts the whole system in danger if any failure happens.

In order to maintain the reliability, the idea is to partially integrate the erasure coding solution so as to take care of the low access potential files. For reference, the storageCore [3] is chosen because of the efficiency in terms of network bandwidth and computation costs. The details of the entire erasure coding process are not the focus of this research and can be found in the original paper [3]. As time progresses, the data files are divided into two sets: the replication set and the erasure set. Only the files in the replication set have their access potentials calculated and replicated over to the system. When the access potential of a file decreases and thus results in its replica quantity being less than or equal to the minimum number of replicas, the status of the file is marked as restricted for replicating and this file is moved to the erasure set. As soon as the file transfer is finished, the erasure coding process begins, encoding this file at the block level. Even though the replication process is fully postponed for the erasure coded file, the minimum replicas of the file are still kept so they can provide access without inducing degradation on the reading. Specially, the erasure coded file still has a chance to return to the replication set if there is any remote access firing for its remaining replicas. When this occurs, the restriction on the original file is simply lifted, enabling replication once more. To deal with any potential failures, the modified HDFS would search the replication set first. If there is no functional copy of the file, the erasure coding reconstruction process is triggered to fix the problem. Due to the nature of the proposed method, it could be considered as a hybrid solution of erasure code and replication.

3.3 Domain analysis

As stated previously, the purpose of this research focuses on pro-actively improving the data locality based on the prediction method. For that reason, it is necessary to discuss the properties of input and output. Intuitively, the predictive computation mostly relies on the heartbeat (the periodic information generated by the computing nodes to indicate their operational status), which is collected by the HDFS logging component. This heartbeat is periodic, noise-free and consists of access rate as well as access type with regards to the time epoch. Basically, there are two kinds of access types: remote access and local access. Local access is dispatched from the tasks on the local machine, while remote access comes from the other servers in the same rack or from the servers located on the different racks, and is also known as the inter-communication access. Since each type of access possesses a very different data transmission rate, it is necessary to consider this rate as a penalty factor that encourages the localization. This factor is introduced later in the Prediction Model section. After the prediction, the result, which comprises the access potential as well as the access pattern, is used for the replication management process.

3.4 Cost analysis

The replication and the data locality are not in a linear relationship [19], only the storage space consumption, the network bandwidth and the disk I/O are linearly related to the replication factor. In other words, the quantitative changes in the replication factors affect the storage cost and the communication bandwidth, which might subsequently degrade the currently running tasks. Unfortunately, this is an unavoidable issue for any solution attempting to improve the data locality and/or the reliability. Frankly, there is no ideal solution to achieve the best functionality without such a trade-off, especially when compared with the state of the art solution. Because of that, our goal is to design an architecture that improves the metric of interest (the data locality) but still maintains the same level of reliability at a reasonable price. For the reliability, as discussed above, the maintenance is held on two sets: the replication set and the erasure set. In fact, the reliability of the replication set is not a problem. First, due to the over replication, the redundancy of this set is kept at a high level as compared with the default scheme. Second, because the replication set occupies less than 10% of total data files [18], the space and transferring cost for this set is much less than that of the triple replication approach. For over 90% of data files which belong to the erasure set, it is proved in the original paper [3] that the reliability of this set is also maintained at a much cheaper cost than the default replication scheme. Given the fact that 80% of data accesses go toward less than 10% of stored bytes, our proposed architecture attempts to identify and replicate this small potential replication set to serve the major aforementioned percentage of data accesses. In this case, a maximum of 10% of data files get replicated with the adaptive factors, which consequently cuts down on the resources used and produces less contention on the currently running tasks as compared with other solutions.

For the configuration of erasure coding component, we decide to use the coding parameter set (n=14, k=12, t=5) where n, k and t stand for the number of stripes in a group, the parity block and the object in the CORE matrix, respectively. Note that the CORE matrix is used to represent the available and the failed nodes of the system. This configuration produces only 20% storage overhead as compared with HDFS-RAID, but consumes 50% less bandwidth and improves up to 76% faster speed for repairing [3]. Later in the Performance Evaluation section, in the TeraSort experiment, given that the input data and the block size are 1TB and 128MB, respectively, the total number of blocks is around 7813. Among this number, only 10% of blocks (approximately 781 blocks) are replicated with the peak replication factor (which is 5.6 in the experiment). The remaining 7032 blocks, which belong to erasure set, are kept with the minimum factor equals to 1. By simple calculation, the total number of replicas is 11406 blocks. In

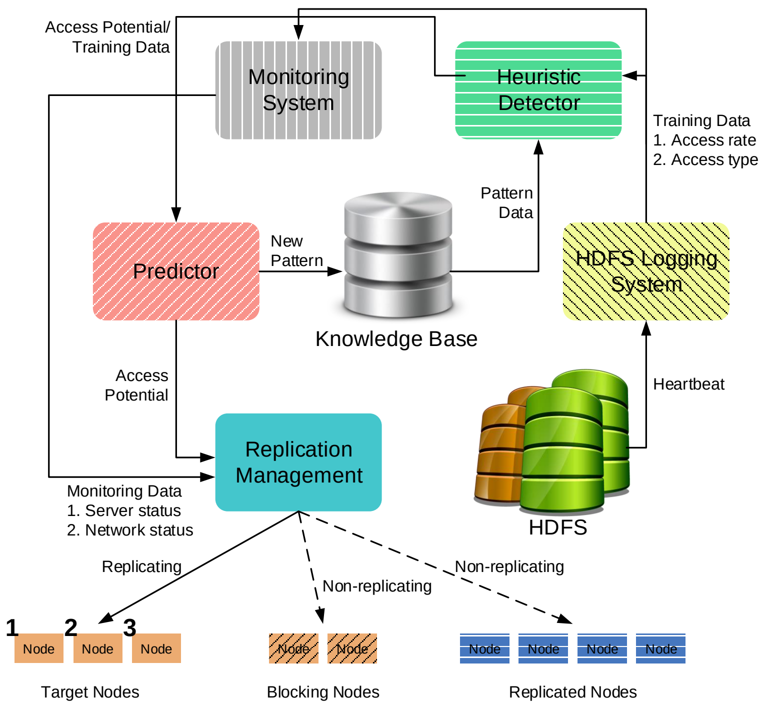


Fig. 1: Architecture of Adaptive Replication Management (ARM) system.

addition, the erasure set roughly creates 4219 blocks (which are 60% of erasure set). It means the storage cost of our approach is 15625 blocks compared with 23439 blocks of default replication scheme (34% less overhead for the storage). This advantage significantly mitigates the overhead on the network bandwidth, the disk operation and the storage resource.

4 Proposed Architecture

4.1 System description

The main function of the proposed architecture is to dynamically scale the replication factors as well as to efficiently schedule the placement of replicas based on the access potential of each data file. Additionally, to reduce the calculation time, the knowledge base and heuristic technique are implemented to detect the similarity in the access pattern between in-processing files and the predicted ones. By definition, the access pattern is actually a set of eigenvectors describing the feature properties of processed data. Two files with similar access behaviors are treated with the same replication strategy. However, because these techniques are minor parts and popularly used in various systems, discussing them is not within the scope of this paper. Constructed as a component of HDFS, the proposed approach (ARM) takes responsibility in managing the replication over the HDFS nodes. Intuitively, an overview of ARM is described in Figure 1. In this architecture, the traditional physical servers as well as the cloud virtual machines can be used as and referred to as nodes. For this system configuration, ARM can be considered as a replication scheduler which can collaborate with any MapReduce job scheduler. In fact, ARM helps the Fair scheduler and delay scheduling algorithm [2] to overcome the drawback of long tasks. Following is the description explaining the operation of ARM.

To begin with, the system starts by periodically collecting the heartbeat. After that, this heartbeat is sent to the heuristic detector as the training data. This training data is compared with the access patterns, which are extracted from the predictor component and stored at the knowledge base. If there is a match, the access potential is then retrieved from the pattern and directly passed to the predictor component without any computation. Otherwise, the training data is continuously sent instead as described in Figure 2. In that case, most of the computation belongs to the hyper-parameter learning and training phases of the prediction. To solve this issue, the hyper generator is constructed to reduce the computational complexity of the hyper-parameter learning phase. After that, the training phase can start to estimate the access potential. Finally, the access potential of the target file is passed on to the replication management component. In addition, a new pattern is also extracted and stored at the knowledge base for the next evaluation.

4.2 Replication management

The purpose of this section is to describe how the replication management chooses the placement for the replica. Theoretically, by placing the potential replicas on low utilization nodes (low blocking rate nodes), the replication management helps to redirect the tasks to these idle nodes and balance the computation. The blocking rate is calculated based on the information provided by the monitoring system. Based on Ganglia framework [20], the monitoring system is simple, robust and easy to configure for monitoring most of the required metrics. After plugging into the HDFS nodes, the monitoring system can collect statistics *via* Ganglia API.

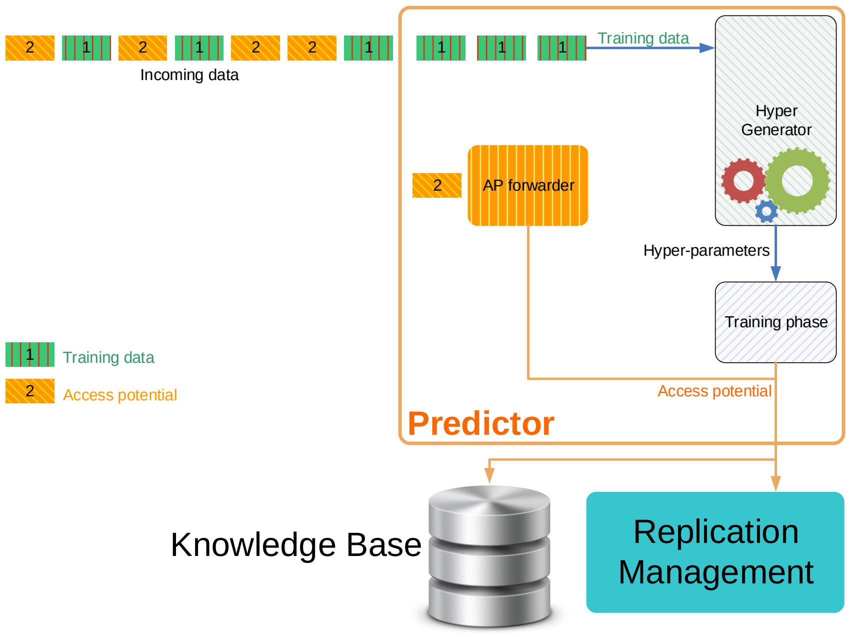


Fig. 2: Working mechanism of Predictor component.

Because Ganglia receives most of the metrics provided by HDFS, there is almost no difference between this statistic and the heartbeat. The only extra information is the system statistic, which consists of CPU utilization, RAM utilization, disk I/O and network bandwidth. This design helps to unify the data sources for computational convenience, especially for blocking rate calculation.

In order to complete the replication management, we assume that the replication management component collects all the ingredients and generates the replication strategies. From this assumption, the access potential is used to scale the number of file copies. Then, the only issue remaining is related to choose the placement of the replicas. As mentioned above, this duty is mainly based on the statistics retrieved from the monitoring system to calculate the blocking rate and assign the replicas. Using the parallel and distributed system theory [21], only a few critical factors can be considered to judge the blocking rate of the server. These factors include the network bandwidth, the number of concurrent accesses and the capability of the server. Following is the mechanism to calculate the blocking rate.

By denoting the limited number of slots of node S_i as c_i, S_i might reach a high blocking rate if all of the slots tend to be occupied by the map tasks. It is worth noting that the improvement of data locality only benefits the map phase, not the shuffle and reduce phases. Therefore, the blocking rate calculation is applied only to the map tasks. The probability of node S_i being fully occupied by the map tasks defines the blocking rate of S_i, which is represented by $\mathcal{BR}(S_i)$. Because the arrival rate λ_i of the map tasks coming to node S_i follows the Poisson distribution, the service process of S_i is considered to be of the $M/M/c$ queuing model. By definition, the $M/M/c$ Markov chain model is a stochastic process in which the first M stands for the Poisson arrival rate of customers, the second M is the exponential service rate of the servers and c generally represents the capacity of each computing node (in this case, c represents the slot capacity). As a result, the blocking rate of S_i follows the Poisson arrival see time averages (PASTA) theory [22] as below:

$$\mathcal{BR}(S_i) = \frac{(\lambda_i \tau_i)^{c_i}}{c_i!} \left[\sum_{k=0}^{c_i} \frac{(\lambda_i \tau_i)^k}{k!} \right]^{-1} \tag{1}$$

where τ_i is the average mapping time of the aforementioned tasks in S_i. Thereafter, by evaluating the blocking rate, it is easy for the replication management component to select a location to assign the replicas. As described in Figure 1, only the computing node satisfying two conditions is chosen as the destination. The first condition is low-blocking rate and the second one is to not store the desired replicas in advance.

5 Prediction Model

5.1 Background

The objective of prediction in the proposed approach (ARM) is to anticipate the access potential of the data file. To obtain this target, the Bayesian learning and Gaussian process are employed as the inference technique and probability framework, respectively. These techniques are supposed to perform on the access rate to create the access potential. For the access type, because of the different impacts between the local access and remote access to the transmission rate, a weighted access scheme should be engaged to benefit the localization.

$$W_i = \sum_j \frac{\eta r_j^{(i)}}{\psi_j a_i} \tag{2}$$

where W_i is the weighted factor corresponding to the access potential of the file i, $\eta = \psi_{max}/\psi_{min}$ is the ratio of the largest to the smallest transmission rate, a_i is the total number of accesses to the file i, $r_j^{(i)}$ is the number of remote accesses coming from the connection j to the file i, and ψ_j is the transmission rate of that connection. The idea of the weighted access scheme is that it reflects the viewpoint that the data files possessing higher remote access rate should have a higher potential to be replicated than the others. By using this factor, not only the access rate, but also the access type can contribute to scale the access potential.

For the access rate, the information provided by this data keeps the critical role in the prediction process. In fact, the access rate actually acts as the input for the whole computation. Basically, this data is comprised of a time location $x = [x_1, x_2, x_3, \cdots x_n]$ and a finite set of random variables $y = [y_1, y_2, y_3, \cdots y_n]$ that represent the corresponding joint Gaussian distribution of the access rate with regards to the time order. This set of time constraints forms the Gaussian process as follows.

$$f(y|x) \sim \mathcal{GP}\big(m(x), k(x, x')\big) \tag{3}$$

with

$$m(x) = \mathbb{E}\big(f(x)\big) \tag{4}$$

$$k(x, x') = \mathbb{E}\bigg(\big(f(x) - m(x)\big)\big(f(x') - m(x')\big)\bigg) \tag{5}$$

where $m(x)$ is the mean function evaluated at the time location variable x and $k(x, x')$ is the positive-definite covariance function [23]. Usually, the square-exponential (SE) kernel, also known as the radial basis function (RBF) kernel, which is chosen to be the covariance function as follows.

$$k_{SE}(x, x') = \sigma_f^2 \exp\bigg(-\frac{(x - x')^2}{2l^2}\bigg) \tag{6}$$

where σ_f is an output-scale amplitude and l is a time-scale of the variable x from one moment to the next. Further, l also plays the role of judgment for automatic relevance detection (ARD) to discard the irrelevant input.

By denoting the incoming value of the input data as x_*, the joint distribution of the training output as y, and the test output as y_*, then the posterior distribution of the Gaussian process is evaluated as follows.

$$p\bigg(\begin{bmatrix} y \\ y_* \end{bmatrix}\bigg) = \mathcal{GP}\bigg(\begin{bmatrix} m(x) \\ m(x_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K}(x, x') & \mathbf{K}(x, x_*) \\ \mathbf{K}(x_*, x) & \mathbf{K}(x_*, x_*) \end{bmatrix}\bigg) \tag{7}$$

Here, $\mathbf{K}(x_*, x_*) = k(x_*, x_*)$ and $\mathbf{K}(x, x_*)$ is the column vector made from $k(x_1, x_*), k(x_2, x_*) \cdots, k(x_n, x_*)$. In addition, $\mathbf{K}(x_*, x) = \mathbf{K}(x, x_*)^\top$ is the transpose of $\mathbf{K}(x, x_*)$. After that, the posterior distribution over y_* can be evaluated with the below mean m_* and covariance C_*:

$$m_* = m(x_*) + \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1}(y - m(x)) \tag{8}$$

$$C_* = \mathbf{K}(x_*, x_*) - \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1}\mathbf{K}(x, x_*) \tag{9}$$

and so

$$p(y_*) \sim \mathcal{GP}(m_*, C_*) \tag{10}$$

The best estimation for the predictive access potential y_* is the mean of this posterior Gaussian distribution, given by

$$\overline{y}_* = \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1}y \tag{11}$$

Also, the uncertainty of the estimation is captured in the variance of the distribution as follows.

$$var(y_*) = \mathbf{K}(x_*, x_*) - \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1}\mathbf{K}(x, x_*) \tag{12}$$

Theoretically, the predictive access potential retrieved from the Gaussian process regression (GPR) is highly accurate compared to other regression methods [24]. However, the standard implementation of GPR costs $O(n^3)$ for computational complexity and $O(n^2)$ for storage complexity when calculating n training points of the dataset [25]. Most of the complexity comes from calculating the matrix inverse and log determinant in both hyper-parameter learning and training phases, which are two main phases of GPR. This can be seen as the drawback of GPR, which prevents the prediction from being quickly calculated on a large dataset. To solve this problem, we introduce the complexity reduction and optimization technique to each phase of the prediction process.

5.2 Hyper-parameter learning phase

Algorithm 1: Hyper-parameter learning phase

```
   Data: Access array. This is the latest history of access
         rate of each data file with regards to time step.
   Result: Hyper-parameters array $\theta^{(*)} = [l^{(*)}, \sigma_f^{(*)}]$
1  Initialize value for $\theta^{(0)} = [l^0, \sigma_f^0], \omega, \epsilon_{RMSE}$;
3  /* Fast Fourier Transform of input data
      */
4  $\hat{y}$= nufft1d1(y);
5  for k=1 to sizeof($\hat{y}$) do
7      /* step_size is equivalent to $\alpha$ in
          the Equation (22) and (23)          */
8      step_size=decay_function($k$);
9      $j$=random(1,sizeof($\hat{y}$));
11     /* partial derivative of $\mathcal{F}_{rMLL}$ w.r.t
          $l$                                    */
12     $\nabla l$ = partial_l($\hat{y}_{[j]}, \omega_{[j]}, l^{(k-1)}, \sigma_f^{(k-1)}$);
14     /* partial derivative of $\mathcal{F}_{rMLL}$ w.r.t
          $\sigma_f$                             */
15     $\nabla\sigma_f$ = partial_$\sigma_f$($\hat{y}_{[j]}, \omega_{[j]}, l^{(k-1)}, \sigma_f^{(k-1)}$);
17     /* update hyper-parameters               */
18     $l^{(k)} = l^{(k-1)}$+step_size*$\nabla l$;
19     $\sigma_f^{(k)} = \sigma_f^{(k-1)}$+step_size*$\nabla\sigma_f$;
20     Compute $\mathcal{F}_{rMLL}^{(k)}(\theta^{(k)})$;
21     Compute $RMSE^{(k)} = RMSE(\mathcal{F}_{rMLL}^{(k)})$;
22     if ($RMSE^{(k)} \leq \epsilon_{RMSE}$) then
23         break();
24     end
25 end
26 return $\theta^{(*)} = [l^{(*)}, \sigma_f^{(*)}]$;
```

Before proceeding with the prediction of the access potential, the GPR model requires the determination of the hyper-parameters as a prerequisite. By definition, hyper-parameters, which can be found in the covariance function, are the free parameters making the adaptations to the prediction process if the dataset at runtime. Finding hyper-parameters is one of the most expensive steps in constructing the aforementioned prediction model. However, not much progress has been made in terms of performance improvement, especially in dealing with a large dataset. In this section, we introduce a method that significantly reduces the complexity of hyper-parameters learning phase, based on a combination of the fast Fourier transform (FFT), the law of log determinant and the stochastic gradient descent (SGD). This enhancement is the first of a two-step process to actualize the possibility of using GPR in predicting the access potentials. Following is the step-by-step explanation of our method.

Denoting the set of hyper-parameters of the covariance function in Equation (6) as $\theta = [\sigma_f, l]$, this set is supposed to be evaluated through the marginalization process. First, rewrite Equation (10) by using Bayes' rule, as shown below:

$$p(y_*|y) = \frac{\int p(y_*|y,\theta)p(y|\theta)p(\theta)\mathrm{d}\theta}{\int p(y|\theta)p(\theta)\,\mathrm{d}\theta} \tag{13}$$

In this equation, the marginal likelihood $p(y) = \int p(y|\theta)p(\theta)\mathrm{d}\theta$ is the main point of interest for identifying θ. So far, the estimation of this hyper-parameter set θ can be obtained when $p(\theta|y)$ reaches its maximum [26]. According to Bayes' rule, $p(\theta|y)$ is known to be proportional to $p(y|\theta)$. Then, the estimation step only involves maximizing $\log p(y|\theta)$ or minimizing the negative of $\log p(y|\theta)$ [27], which is shown below:

$$-\log p(y|\theta) = \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} + \frac{1}{2}\log|\mathbf{K}| + \frac{n}{2}\log(2\pi) \tag{14}$$

Due to the high complexity of this calculation, represented by the matrix inverse $\mathbf{K}^{-1}$, finding an alternative method for the hyper-parameter learning process is necessary. Instead of putting effort into minimizing Equation (14), the heavy-load-job can be done more quickly by approximately minimizing the upper bound of this term. Analytically, the dominant computation arising from Equation (14) are focused on two terms: the data-fit term [24] denoted by $\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y}$ and the log determinant $\log|\mathbf{K}|$. These terms can be simplified to reduce the complexity. To do that, the law of log determinant [28] is firstly used. Initially, by calculating the log determinant $\hat{L}$ of the sample covariance matrix $\hat{\mathbf{K}}$, Equation (14) is simplified to

$$-\log p(y|\theta) = \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} + \frac{1}{2}\hat{L} + \frac{n}{2}\log(2\pi) \tag{15}$$

After a number of calculations, the term $\hat{L}$ converges to a constant. This convergence leads to the conclusion that minimizing the negative log marginal likelihood in this domain might only involve minimizing the following reduced negative marginal log likelihood (rMLL).

$$-\log p(y|\theta)_{rMLL} = \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} \tag{16}$$

Traditionally, dealing with this task concerns finding the inverse of the covariance matrix $\mathbf{K}$, which is very computationally expensive. To mitigate this issue, a Fourier transform can be applied. As mentioned before, because the covariance function is positive-definite, it is possible to do the transformation by using the Fourier transform to bring the computation from the spatial-temporal domain into the periodic domain. Consequently, the cost reduces to $O(nlogn)$ when calculating n training points of dataset. Obviously, this cost is much more preferred and faster than the traditional computation.

In order to achieve the above improvement, the squared exponential kernel $k_{SE}(x, x')$ in Equation (6) needs to be re-written in the Fourier transform representation [29] as shown below:

$$\mathcal{F}_{SE}(\omega) = l\sigma_f^2\sqrt{2\pi}exp(-2\pi^2\omega^2 l^2) \tag{17}$$

where, ω is the frequency representation of the time location x in the periodic domain. Assume that $\mathbf{\Phi}$ is the function that generates $\tilde{\mathbf{K}} = \mathbf{K}^{-1}$. Under the periodic domain, the Parseval theorem [30] is applied to derive the Fourier transform for Equation (16) as follows.

$$\mathcal{F}_{rMLL}(\theta) = \mathcal{F}\big(-\log p(y|\theta)_{rMLL}\big) = \frac{1}{2n}\hat{\mathbf{y}}^T\widehat{\mathbf{\Phi} * \mathbf{y}_\circ} \tag{18}$$

where the 'hat' sign above $\hat{\mathbf{y}}$ denotes the Fourier transform of $\mathbf{y}$, and $\mathbf{y}_\circ$ denotes the data vector in the periodic domain. In the next step, by continuously applying the convolution theorem with regards to the constraint $\Phi\mathcal{F}_{SE} \equiv 1$, the Fourier transform of rMLL can be represented as follows.

$$\mathcal{F}_{rMLL}(\theta) = \frac{1}{2n}\sum_i \hat{\Phi}_i * \hat{y}_i^2 = \frac{1}{2n}\sum_i \frac{\hat{y}_i^2}{\mathcal{F}_{SE}(\omega_i)} \tag{19}$$

With this form of Equation (19), the set of hyper-parameters θ can be estimated by using gradient-based techniques. In this research, the stochastic gradient descent (SGD) is chosen because of its features of fast convergence and less sensitive to the local minima [31]. To integrate the SGD, the partial derivatives of the Equation (19) are required for each hyper-parameter. These equations are given by

$$\frac{\partial}{\partial l}\mathcal{F}_{rMLL} = \hat{y}_i^2 \exp\left(2\pi^2 l^2\omega^2\right)\left(\frac{2\sqrt{2}\pi^{3/2}\omega^2}{\sigma_f^2} - \frac{1}{\sqrt{2\pi}l^2\sigma_f^2}\right) \tag{20}$$

and

$$\frac{\partial}{\partial \sigma_f}\mathcal{F}_{rMLL} = -\frac{\sqrt{\frac{2}{\pi}}\hat{y}_i^2\exp\left(2\pi^2 l^2\omega^2\right)}{l\sigma_f^3} \tag{21}$$

After that, an updating scheme is issued to update the hyper-parameters to the convergent point. This scheme is as follows.

$$l^{(k)} \leftarrow l^{(k-1)} + \alpha(k)\frac{\partial}{\partial l^{(k-1)}}\mathcal{F}_{rMLL} \tag{22}$$

$$\sigma_f^{(k)} \leftarrow \sigma_f^{(k-1)} + \alpha(k)\frac{\partial}{\partial \sigma_f^{(k-1)}}\mathcal{F}_{rMLL} \tag{23}$$

where $\alpha(k) = {}^{1}/_{(k+1)}$ is the Robbins-Monroe decay function with regards to the k^{th} iteration. This function is chosen instead of the exact line search or the backtracking line search [31], mainly because of the performance issue.

To govern the number of iterations, an error function is defined based on the root mean square error (RMSE) method to measure the convergence. Note that the RMSE method is stricter than the popularly-used mean square error (MSE) method. By using this error function, the gap between the current iteration value and the previous one can be evaluated as follows.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\mathcal{F}_i^{(k)} - \mathcal{F}_i^{(k-1)}}{n}} \tag{24}$$

where, $\mathcal{F}_i^{(k)}$ and $\mathcal{F}_i^{(k-1)}$ stand for the values of rMLL at the target location i in the k^{th} and $(k-1)^{th}$ iterations,

TABLE 1: Computation cost of proposed method

	Direct Method	Conjugate Gradient	Proposed Method
Hyper-parameters learning	$O(n^3)$	$O(n^2)$	$O(nlogn)$
Training	$O(n^3)$	$O(n^2)$	$O(n)$

respectively. Theoretically, the RMSE threshold is limited to 10^{-11} which produces a solution close to the real one. From this moment, the computation can be done without doing the matrix inverse. By the end of this hyper-parameter learning phase, the set of hyper-parameters is ready for the training phase of the predictor. The algorithm of the hyper-parameter learning phase can be found in Algorithm 1.

5.3 Training phase

After having the hyper-parameters, the predictor can start finding the access potential *via* estimating the mean value of the Equation (11). This step is called as the training phase of the prediction. In this step, once again the matrix inverse represented by $\mathbf{K}(x, x')^{-1}$ becomes the main obstacle. Although the FFT is still effective in this case, a more appropriate technique is applied for better performance.

At first, the Equation (11) can be re-written as follows.

$$\overline{y}_* = \mathbf{K}(x_*, x)\xi \tag{25}$$

in which

$$\xi = \mathbf{K}(x, x')^{-1}y \tag{26}$$

multiplying both sides of Equation (26) by $\mathbf{K}(x, x')$, we have

$$y = \mathbf{K}(x, x')\xi \tag{27}$$

Since the matrix $\mathbf{K}(x, x)$ is symmetric and positive-definite, the conjugate gradient [32] iterative method is engaged. The definition of this method is to choose a starting point and make a series of iterative steps that converge upon the approximate solution ξ_i, which is adjacent to the real solution ξ. The best solution for this process can be given by the inequality below:

$$\frac{\|\xi - \xi_i\|_{\mathbf{K}}}{\|\xi - \xi_0\|_{\mathbf{K}}} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2i} \tag{28}$$

where the constant $\kappa = \lambda_{max}/\lambda_{min}$ is the ratio of the largest and the smallest eigenvalues of the matrix $\mathbf{K}$, and the $\mathbf{K}-norm$ is calculated as $\|z\|_{\mathbf{K}} = z^T\mathbf{K}z$ where z is any arbitrary vector. The tolerance parameter ζ is also given such that $0 < \zeta < 1$. This parameter ζ acts as a bound for the practical conjugate-gradient scheme as follows.

$$\frac{\|y - \mathbf{K}\xi_i\|_2}{\|y - \mathbf{K}\xi_0\|_2} \leq \zeta \tag{29}$$

Note that, at the end of the i^{th} iteration, $\|y - \mathbf{K}\xi_i\|_2$ is obtained as the residual in the Euclidean$-norm$.

The complexity of the conjugate gradient method is $O(in^2)$, in which i is the number of iterations and n is the number of training points. Usually, the system works on an enormous number of training sets, and so the quadratic complexity algorithm might rapidly deteriorate the overall performance. Therefore, the conjugate gradient needs to be coupled with the improved fast Gauss transform (IFGT) method [33] [34] to achieve faster calculation. The IFGT technique is actually derived from the fast Gauss transform (FGT) [35] [36] which is an $\epsilon - exact$ approximate algorithm. To do this integration, in the j^{th} step of the conjugate gradient, the FGT is used to expands the j^{th} conjugacy $G(x_j)$ into a plane-wave expansion of the previous SE kernel as shown below:

$$G(x_j) = \sum_{i=1}^{N} q_i \exp(-\frac{\|x_j - x_i\|^2}{2l^2}) \tag{30}$$

where m is the number of target points and n is the number of source points, x_j is the target point with $\{x_j \in \mathbb{R}^2\}_{j=1,\ldots,m}$, q_i is the source weight with $\{q_i \in \mathbb{R}\}_{i=1,\ldots,n}$, x_i is the source point with $\{x_i \in \mathbb{R}^2\}_{i=1,\ldots,n}$, and l is the bandwidth with $\{l \in \mathbf{R}^+\}$. This Gaussian-type expansion might be calculated approximately by using the discrete Fourier transform [36] [37] as follows.

$$G(x_j) \approx \sum_{|\alpha| \leq p} \mathcal{F}(\alpha) w_\alpha \exp(\frac{i\alpha L(x_j - x_i)}{\sqrt{2}pl}) \tag{31}$$

with $\mathcal{F}(\alpha)$ and w_α are given by

$$\mathcal{F}(\alpha) = \frac{1}{2^3\sqrt{\pi}} \exp(-\frac{L^2|\alpha|^2}{4p^2}) \tag{32}$$

$$w_\alpha = \left(\frac{L}{p}\right)^2 \sum_{y \in U} f(y) \exp(\frac{i\alpha L(c^U - y)}{\sqrt{2}pl}) \tag{33}$$

where $\alpha = (\alpha_1, \alpha_2)$ is a multi-dimensional index set, p is the number of plane-wave coefficient required per dimension to obtain the desired precision ϵ, and L is the truncation error term (the detail configurations of p and L can be found in [37]). Assuming that the domain Ω is a unit square $[0, 1]^2$ (if the value stays out of this range, shifting and rescaling have to be performed), by partitioning the domain Ω into a number of uniform squares U of size $\sqrt{2}l$, the FGT can compute the desired result in three steps: **S2W**, **W2L** and **L2T**. Before explaining these terms, the definition of 'interaction list' should be stated. Briefly, this list is denoted by $\mathcal{I}[U]$ which describes a specific set of a neighborhood of U. Basically, this set supports the kernel at the center of U. At the beginning, the FGT algorithm starts with the **S2W** step. This step sequentially calculates the Equation (31) for each square U. After that, the plane-wave expansion, which is created in **S2W**, now propagates to all elements V of $\mathcal{I}[U]$ as a 'local' expansion. Intuitively, the visualization of propagation steps can be found in Figure 3. In this propagation, the step **W2L** plays the role of modifying the below expansion:

$$w^*_\alpha = w_\alpha \exp(\frac{i\alpha L(c^V - c^U)}{\sqrt{2}pl}) \tag{34}$$

In the last step, **L2T**, the conjugacy $G(x_j)$ is computed at x_j by using the 'local' expansion from the box containing it.

$$G(x_j) = \sum_{|\alpha| \leq p} \mathcal{F}(\alpha) w_\alpha^* \exp\left(\frac{i\alpha L(x - c^V)}{\sqrt{2}pl}\right) \tag{35}$$

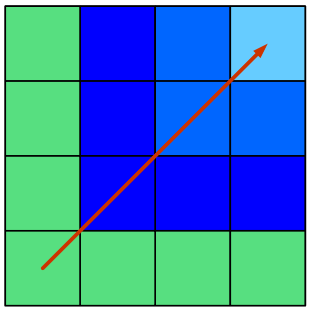


Fig. 3: The 'local' expansion step **S2W** is calculated directly in the green layer. Other layers in the interaction list $\mathcal{I}[U]$ are calculated by the propagation. The direction and the order of the propagation are shown in red arrow and the light magnitude of blue color, respectively.

As mentioned above, the FGT helps to nicely reduce the computational cost to $O(mn)$ where m is the number of target points and n is the number of source points. However, this technique has a reduction in accuracy as a side effect. It is mainly related to the ϵ parameter, which also critically influences the parameters p and L. To overcome this drawback, the IFGT [34] is proposed as a strategy to adaptively select this ϵ parameter without a loss of accuracy as follows.

$$\epsilon_i \leq \frac{\delta}{n} \frac{\|y - K\xi_0\|}{\|\check{r}_{i-1}\|} \tag{36}$$

where δ is the bound of the difference between the residual of the i^{th} iteration and the corresponding residual of the approximate matrix-vector product: $\|\check{r}_i - r_i\| \leq \delta$, and n is the number of training points. With this improvement, the complexity in the training phase drops to $O(n)$ [34].

Although it is suitable for the training phase to adopt the IFGT coupling with conjugate gradient (IFGT-CG), this combination technique is difficult to apply in the hyper-parameter learning phase. The IFGT, which is derived from the FGT technique, works properly only if the objective function can be represented in a Gaussian type form, which is impossible for rMLL in Equation (16). In this case, the standalone conjugate gradient solves the matrix inverse within a computational complexity of $O(n^2)$. This improvement is actually worse than the $O(nlogn)$ of the proposed technique introduced in the previous section. The comparison of the complexity between the proposed method and the others can be found in Table 1.

6 Performance Evaluation

6.1 Experiments

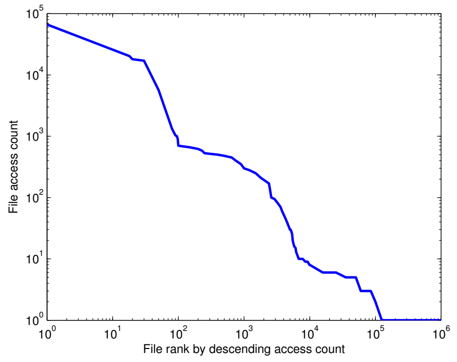
Two experiments are used to evaluate the performance of the proposed approach (ARM). The first experiment is conducted on the Facebook cluster traces, namely the Statistical Workload Injector for MapReduce (SWIM) [38] [18]. By sampling the historical MapReduce cluster traces, the SWIM provides an efficient method to measure the effectiveness of the solution, which is intended to improve the HDFS replication on the realistic dataset. Two sets of synthesized day-long workloads, namely Facebook_trace_01 (*FB-2009*) and Facebook_trace_02 (*FB-2010*) are studied. Each set contains 24 historical traces sampled on a 600-machines cluster. It is worth noting that these workload replay scripts are modified to generate the real-life workloads. In other words, the data access patterns in these sets of traces are adjusted to follow the Zipf-like distribution, *i.e.*, over 90% of data accesses focus on less than 10% of data files as mentioned in the Cost analysis section. The distribution of the file access rate can be found in Figure $4a$. The second experiment is the famous TeraSort stress test released by Yahoo!. This is a benchmark program written in MapReduce and included in the Hadoop distribution by default. Basically, the TeraSort stress test builds a sample key structure by selecting the subsets from the input before submitting the job and pushing this key structure into HDFS. In this experiment, the TeraSort is performed on 1TB input data. Additionally, the Hadoop version used in the experiments is also modified to accept the dynamic replication factor as well as the flexible placement decision which is made by the ARM system. Finally, the configuration of the experimental cluster used for the experiments can be found in Table 2.

TABLE 2: System Configuration

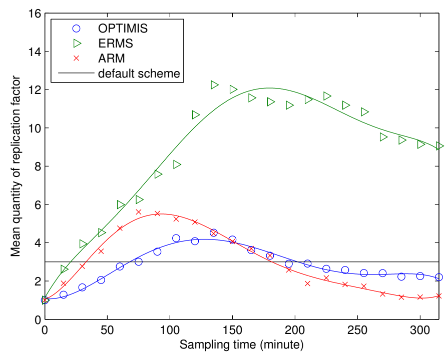
	Configuration
Computing Nodes	01 Name Node, 16 Data Nodes
Platform	64bit
CPU	Intel®Core™ Xeon E5520 2.3GHz 4 cores
Storage	4x1TB for Name Node 4x1TB for each Data Node
Memory	16GB for Name Node 12GB for each Data Node
Network	Gigabit Ethernet Controller
OS	CentOS 6.5 (final) Kernel: 2.6.32-431.el6.x86_64
Software	Apache Hadoop 0.20-append

6.2 Implementation

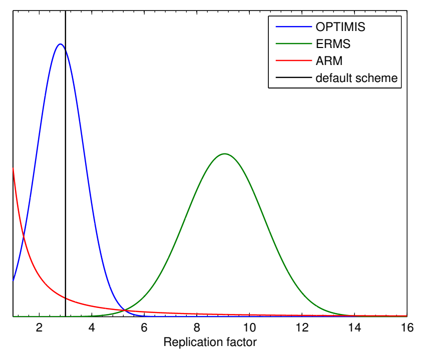
Since there is no way to directly trigger the operations on files in HDFS, an indirect heartbeat analysis is chosen as the alternative technique to collect the read/write operations. To do that, a log parsing script is implemented in *Java* by using the socket listener to manipulate the log file. Basically, the needed information consists of access time, access IP, target file, file operation and user authority who issues the operation. For the comparison purpose, the default replication mechanism, ERMS [9] and OPTIMIS [5] are implemented to compare with the proposed ARM. All these approaches are implemented along with the Fair scheduler. Note that the delay scheduling algorithm is not engaged to avoid confusing the experiment result. Separately, the delay scheduling is subsequently compared with ARM *via* the test on various durations of task, which is described in Figure $4g$. At the prediction level, the proposed prediction method



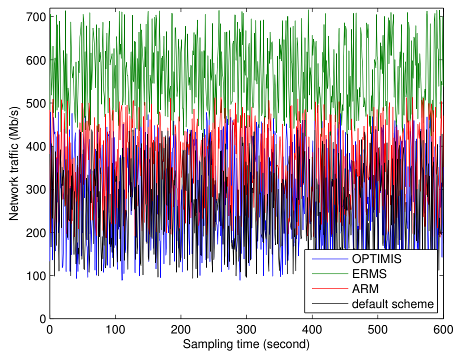
(a) Access rate vs rank.



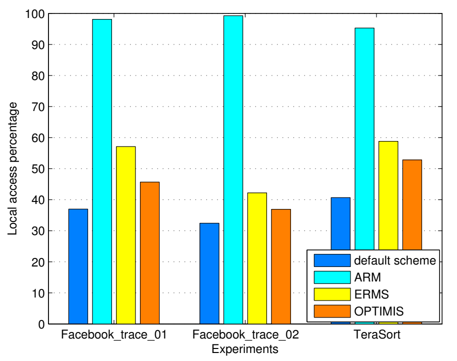
(b) Data redundancy metric evaluation (lower is better).



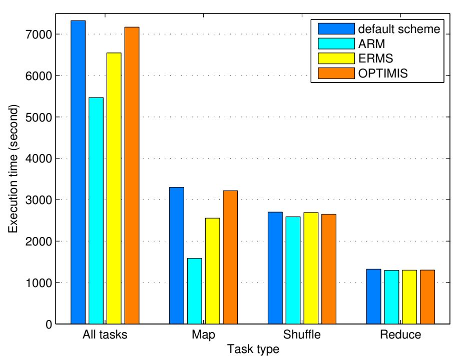
(c) Distribution (pdf) of each method.



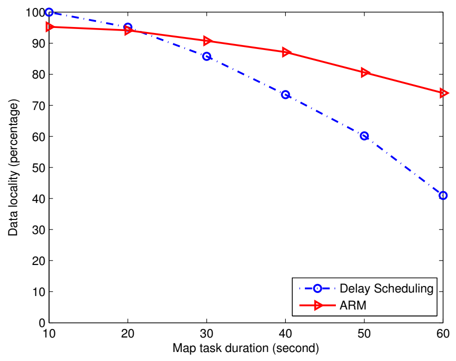
(d) Network traffic evaluation (lower is better).



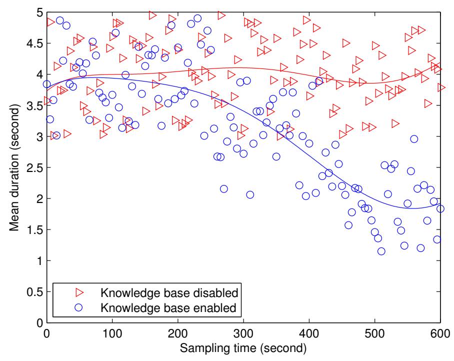
(e) Data locality metric evaluation (higher is better).



(f) Execution time benchmark in the TeraSort stress test (lower is better).



(g) ARM vs delay scheduling in the data locality test over various task durations.



(h) Evaluation on mean duration of creating the access potential (lower is better).

Fig. 4: Performance evaluation of proposed method at system level.

is compared with the conjugate gradient (CG) and the direct method (Gauss-Jordan elimination).

6.3 Metrics

ARM is measured at two levels: the prediction level and the system level. At the prediction level, the metrics of interest are the completion time and the accuracy of prediction. Meanwhile, at the system level, the data redundancy and the availability (measured by the data locality) are the metrics of interest. In addition, some relevant factors such as network traffic, execution time and utilization of knowledge base are also considered. For the data locality evaluation, the metric is calculated in percentage as follows.

$$\mathcal{M}_{dl} = \frac{Access_{local}}{Access_{total}} \tag{37}$$

where $\mathcal{M}_{dl}$ is the data locality metric of the target file estimated by the fraction of the local access $Access_{local}$ and the total access $Access_{total}$. In the other hand, the data redundancy metric is evaluated *via* the mean quantity of replicas and the shape of replication factor distributions.

6.4 Results

System level - data redundancy: this metric is measured in the modified SWIM experiment. By running the modified SWIM for 315 minutes and calculating the mean quantity of the replication factors (Figure $4b$), it can be seen that the average replication factor (1.22) of ARM is even better than the default scheme (3.0). This fact happens because only the files from the replication set (which possesses less than 10% of data files) are gradually assigned the replication schemes. Theoretically, the replication set creates a small number but elite replicas to serve the data access. In the other hand, the remaining low potential files are handled by the erasure set. It is worth noting that the effect and the configuration of the erasure coding part can be found in the previous Cost analysis section. Besides, the improvement of ARM can be seen also in the replication factor distributions (Figure $4c$). In this measurement, the variances of ERMS and OPTIMIS are narrow. It means that the difference between the high and low access potential files is not clearly distinguished. As a consequence, without the knowledge of access pattern, the issue of high network utilization (39.96% higher than ARM) happens to ERMS in effort to improve the data locality metric by declining the thresholds (Figure $4d$). In the other hand, although OPTIMIS does not have the problem of high network traffic because of better access pattern utilization, this approach faces the problem of inaccuracy of predicting technique [5]. This critical issue leads to the dysfunctional replication process which then limits the data locality.

System level - availability (data locality): by averaging the results from both Facebook trace tests, ARM scores approximately 2.84 times better than the default scheme, 1.98 times better than ERMS and 2.39 times better than OPTIMIS (Figure $4e$). In the TeraSort experiment (also in Figure $4e$), ARM achieves 2.34, 1.62 and 1.80 times better than the default scheme, ERMS, and OPTIMIS, respectively. The effectiveness of the data locality improvement can be observed in the execution time benchmark of the TeraSort experiment (Figure $4f$). Firstly, it is important to mention that the data locality benefits the map phase rather than the shuffle and reduce phases. Therefore, in the TeraSort test, ARM helps to reduce 52% and 37.98% mapping time, respectively, as compared with the default replication scheme and the second best approach (ERMS).

As mentioned previously, because the delay scheduling algorithm (hereinafter: delay_sched) might confuse the evaluation result, this algorithm should be compared with ARM in a separate TeraSort test, which is shown in Figure $4g$. Theoretically, the efficiency of the delay_sched decreases with the increment in the map task length [2]. Indeed, the delay_sched achieves 4.71% better than ARM at the data locality measurement when the map task is 10s long. However, when the map task is 22s long, the delay_sched and the ARM score analogous results. By increasing the map task length continuously, the delay_sched is suffered for the decline in data locality metric faster than ARM. By the end of the experiment, when the map task is 60s long, the data locality of ARM is still maintained at 73.96% as compared with 40.98% of the delay_sched. This is the significant advantage of proposed approach over the state of the art method.

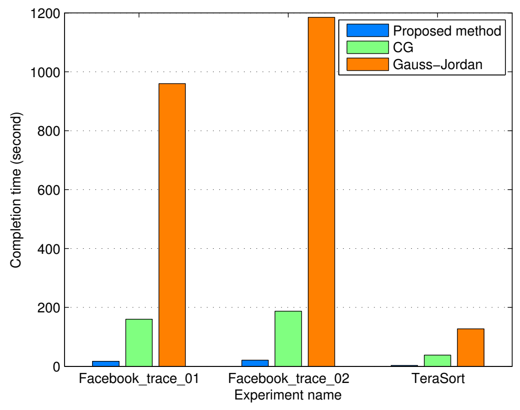
Lastly, the evaluation on the effect of knowledge base is conducted on the TeraSort experiment in Figure $4h$. When getting enough information (after around 250 seconds), the knowledge base and the heuristic detector can reduce the duration of creating the access potential, up to 2.17 times. Frankly, this result is varied as time progresses, but it indicates that caching the access patterns can help to improve the performance of the access potential prediction.

Prediction level - completion time: in Facebook trace experiments, Figure $5a$ shows that within the same error bound ($\epsilon = 10^{-11}$) and the same size of dataset (10^3 points), the proposed method approximately cuts down an amount of hyper-parameters learning time up to 89.37% as compared with the second best method. Because of the approval problem of Facebook trace data, the training phase evaluation is skipped for this kind of experiments. In the TeraSort experiment, as also shown in Figure $5a$, with the different dataset (10^2 target points), the proposed method is measured to be 92.11% faster than the second best method in the hyper-parameters learning phase. It is worth noting that the Gauss-Jordan elimination method takes too much time to finish the tasks, this method is eliminated for later evaluation. Finally, during the training phase as well as the overall prediction evaluation described in Figures $5b$ and $5c$, respectively, when the number of training points increases, the proposed method continues to outperform the other methods in terms of reaction rate.

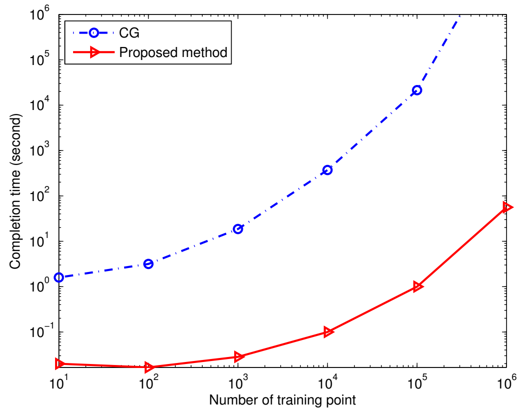
Prediction level - accuracy: depending on the degree of reliability, because the proposed method also partially relies on the IFGT which defines the precision $\epsilon = 10^{-11}$ in advance, the accuracy is safely maintained. Figure $5d$ shows that for an accuracy benchmark of 18 consecutive testing points in the TeraSort experiment, the mean prediction adapts quite well to the testing data with 95% confidence maintained by the variance.

7 Conclusion

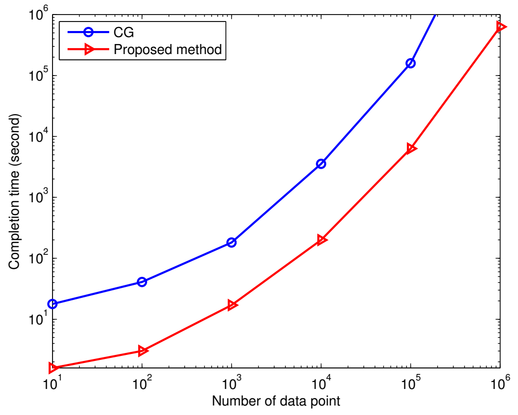
In order to improve the availability of HDFS by enhancing the data locality, our contribution focus on following points.



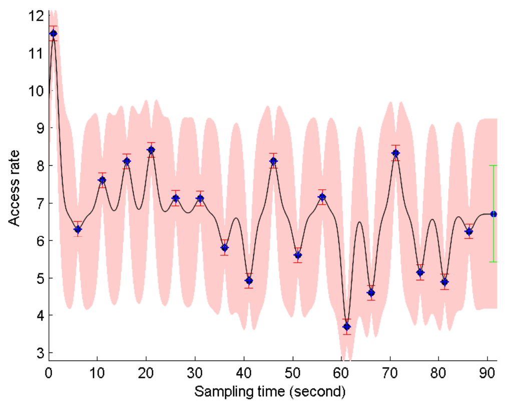
(a) Hyper-parameters learning speed evaluation (lower is better).



(b) Training speed evaluation in the TeraSort benchmark (lower is better).



(c) Overall prediction speed evaluation (lower is better).



(d) Mean prediction and error bar of proposed method given 18 training points in the TeraSort benchmark.

Fig. 5: Performance evaluation of proposed method at prediction level

First, we design the replication management system which is truly adaptive to the characteristic of the data access pattern. The approach not only pro-actively performs the replication in the predictive manner, but also maintain the reliability by applying the erasure coding approach. Second, we propose a complexity reduction method to solve the performance issue of the prediction technique. In fact, this complexity reduction method significantly accelerates the prediction process of the access potential estimation. Finally, we implement our method on a real cluster and verify the effectiveness of the proposed approach. With a rigorous analysis on the characteristics of the file operations in HDFS, our uniqueness is to create an adaptive solution to advance the Hadoop system. For further development, some parts of the source code developed to test our idea would be made available under the terms of the GNU general public license (GPL).

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) NRF-2014R1A2A2A01003914. Corresponding author is Professor Sungyoung Lee (sylee@oslab.khu.ac.kr).

References

[1] “What is apache hadoop?” https://hadoop.apache.org/, accessed: 2015-08-13.

[2] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling,” in *Proceedings of the 5th European conference on Computer systems*. ACM, 2010, pp. 265–278.

[3] K. S. Esmaili, L. Pamies-Juarez, and A. Datta, “The core storage primitive: Cross-object redundancy for efficient data repair & access in erasure coded storage,” *arXiv preprint arXiv:1302.5192*, 2013.

[4] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, and E. Harris, “Scarlett: Coping with skewed content popularity in mapreduce clusters.” in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys ’11. New York, NY, USA: ACM, 2011, pp. 287–300. [Online]. Available: http://doi.acm.org/10.1145/1966445.1966472

[5] G. Kousiouris, G. Vafiadis, and T. Varvarigou, “Enabling proactive data management in virtualized hadoop clusters based on predicted data activity patterns.” in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*, Oct 2013, pp. 1–8.

[6] A. Papoulis, *Signal analysis*. McGraw-Hill, 1977, vol. 191.

[7] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, and D. Feng, “Cdrm: A cost-effective dynamic replication management scheme for cloud storage cluster.” in *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*, Sept 2010, pp. 188–196.

[8] C. L. Abad, Y. Lu, and R. H. Campbell, “Dare: Adaptive data replication for efficient cluster scheduling.” in *CLUSTER*. IEEE, 2011, pp. 159–168.

[9] Z. Cheng, Z. Luan, Y. Meng, Y. Xu, D. Qian, A. Roy, N. Zhang, and G. Guan, "Erms: An elastic replication management system for hdfs." in *Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on*, Sept 2012, pp. 32–40.

[10] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," in *Proceedings of the VLDB Endowment*, vol. 6, no. 5. VLDB Endowment, 2013, pp. 325–336.

[11] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75–86, 2008.

[12] A. Duminuco and E. Biersack, "Hierarchical codes: How to make erasure codes attractive for peer-to-peer storage systems," in *Peer-to-Peer Computing, 2008. P2P'08. Eighth International Conference on*. IEEE, 2008, pp. 89–98.

[13] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci *et al.*, "Windows azure storage: a highly available cloud storage service with strong consistency," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 143–157.

[14] B. Fan, W. Tantisiriroj, L. Xiao, and G. Gibson, "Diskreduce: Replication as a prelude to erasure coding in data-intensive scalable computing," *Parallel Data Laboratory, Carnegie Mellon University, Pittsburgh*, 2011.

[15] S. B. Wicker and V. K. Bhargava, *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[16] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," *ACM Transactions on Storage (TOS)*, vol. 9, no. 1, p. 3, 2013.

[17] A. Datta and F. Oggier, "Redundantly grouped cross-object coding for repairable storage," in *Proceedings of the Asia-Pacific Workshop on Systems*. ACM, 2012, p. 2.

[18] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1802–1813, 2012.

[19] Z. Guo, G. Fox, and M. Zhou, "Investigation of data locality in mapreduce," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*. IEEE Computer Society, 2012, pp. 419–426.

[20] "What is ganglia?" http://ganglia.sourceforge.net/, accessed: 2015-08-13.

[21] X. Wu, *Performance Evaluation, Prediction and Visualization of Parallel Systems*, ser. The International Series on Asian Studies in Computer and Information Science. Springer US, 1999. [Online]. Available: http://books.google.co.kr/books?id=IJZt5H6R8OIC

[22] R. Gallager, *Stochastic Processes: Theory for Applications*. Cambridge University Press, 2013. [Online]. Available: http://books.google.co.kr/books?id=CGFbAgAAQBAJ

[23] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms." *Neural Networks, IEEE Transactions on*, vol. 12, no. 2, pp. 181–201, Mar 2001.

[24] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive Computation And Machine Learning. MIT Press, 2005. [Online]. Available: http://www.gaussianprocess.org/gpml/chapters/

[25] K. Chalupka, C. K. I. Williams, and I. Murray, "A framework for evaluating approximation methods for gaussian process regression." *CoRR*, vol. abs/1205.6326, 2012.

[26] E. G. Tsionas, "Maximum likelihood estimation of stochastic frontier models by the fourier transform." *Journal of Econometrics*, vol. 170, no. 1, pp. 234–248, 2012.

[27] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kgl, "Algorithms for hyper-parameter optimization." in *NIPS*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, Eds., 2011, pp. 2546–2554.

[28] T. T. Cai, T. Liang, and H. H. Zhou, "Law of log determinant of sample covariance matrix and optimal estimation of differential entropy for high-dimensional gaussian distributions." *CoRR*, vol. abs/1309.0482, 2013.

[29] P. Sollich and C. K. I. Williams, "Understanding gaussian process regression using the equivalent kernel." in *Deterministic and Statistical Methods in Machine Learning*, ser. Lecture Notes in Computer Science, J. Winkler, M. Niranjan, and N. D. Lawrence, Eds., vol. 3635. Springer, 2004, pp. 211–228. [Online]. Available: http://dblp.uni-trier.de/db/conf/dsmml/dsmml2004.html

[30] J. de Baar, R. Dwight, and H. Bijl, "Speeding up kriging through fast estimation of the hyperparameters in the frequency-domain." *Computers & Geosciences*, vol. 54, no. 0, pp. 99–106, 2013.

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. [Online]. Available: http://books.google.co.kr/books?id=mYm0bLd3fcoC

[32] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain." Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CMU-CS-94-125, 1994.

[33] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis, "Improved fast gauss transform and efficient kernel density estimation." in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Oct 2003, pp. 664–671 vol.1.

[34] C. Yang, R. Duraiswami, and L. S. Davis, "Efficient kernel machines using the improved fast gauss transform," in *Advances in neural information processing systems*, 2004, pp. 1561–1568.

[35] T. I. Alecu, S. Voloshynovskiy, and T. Pun, "The gaussian transform." in *EUSIPCO2005, 13th European Signal Processing Conference*, 2005, pp. 4–8.

[36] L. Greengard and J. Strain, "The fast gauss transform." *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 1, pp. 79–94, 1991.

[37] M. Spivak, S. K. Veerapaneni, and L. Greengard, "The fast generalized gauss transform." *SIAM J. Sci. Comput.*, vol. 32, no. 5, pp. 3092–3107, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1137/100790744

[38] Y. Chen, A. Ganapathi, R. Griffith, and R. H. Katz, "The case for evaluating mapreduce performance using workload suites." in *MASCOTS*. IEEE, 2011, pp. 390–399.

Dinh-Mao Bui received the B.S. degree in Computer Science from the Computer Engineering Department at Ton Duc Thang University, Vietnam, in 2009 and the M.S. degree in Data Communication and Networking from the Posts and Telecommunications Institute of Technology, Vietnam, in 2012. He is now working toward the PhD degree in the Department of Computer Engineering at Kyung Hee University, Korea. His research interests include Convex Optimization, Stochastic Process and Big Data.

Shujaat Hussain received his B.S. with distinction from Mohammad Ali Jinnah University University, Pakistan in 2005. He got M.S. from Mohammad Ali Jinnah University University, Pakistan in 2009. Since September 2011, he has been working on his PhD degree in the Department of Computer Engineering at Kyung Hee University, Korea. His research interests include cloud computing, health analytics in big data and distributed systems.

Eui-Nam Huh is a chair professor in Dept. of Computer Engineering. His interesting research areas are: Cloud Computing, Big Data Computing, IoT, Distributed Real Time System, Network Security. He earned Masters degree in Computer Science from University of Texas, USA in 1995 and Ph. D degree from the Ohio University, USA in 2002. He has also served for the WPDRTS/IPDPS, SOICT, APIST, ICUIMC and ICCSA community as various chair positions since 2003.

Sungyoung Lee received his Ph.D. degree in Computer Science from Illinois Institute of Technology (IIT), Chicago, Illinois, USA in 1991. He has been a professor in the Department of Computer Engineering, Kyung Hee University, Korea since 1993. Before joining Kyung Hee University, he was an assistant professor in the Department of Computer Science, Governors State University, Illinois, USA from 1992 to 1993. His current research focuses on Ubiquitous Computing, Cloud Computing, Intelligent Computing and eHealth.