2015 Third International Conference on Advanced Cloud and Big Data

# Replication Management Framework for HDFS based on Prediction Technique

Dinh-Mao Bui*, Thien Huynh-The*, Sungyoung Lee*, Bin Li†, Jin Wang†
*Computer Engineering Department, Kyung Hee University, Suwon, Korea
{mao, thienht, sylee}@oslab.khu.ac.kr
† College of Information Engineering, Yangzhou University, China
{lb}@yzu.edu.cn
† College of Information Engineering, Yangzhou University, China
{jinwang}@yzu.edu.cn

***Abstract*—The number of application based on Apache Hadoop is increasing dramatically due to the robustness and dynamic features of this system. At the heart of Apache Hadoop, the Hadoop File System (HDFS) provides the reliability, scalability and high availability to computation by applying a static replication strategy. However, because of the characteristics of parallel operations on the application layer, the accessing frequency for each data file in HDFS is totally different. Consequently, maintaining the same replicating mechanism for every data file might lead to bad effects on the performance. By rigorously considering the drawbacks of HDFS architecture, 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 an individual replication strategy. Eventually, the data file can subsequently be replicated depending on its own access potential. Hence, this approach simultaneously improves the data locality while keeping the analogous redundancy of data storage in comparison with the default replicating scheme.**

***Keywords*—Replication, HDFS, proactive prediction, Bayesian Learning, Gaussian Process.**

## I. Introduction

Evolution on Big Data has created trends in application and solution development to extract, process, and store useful information as it emerges to deal with new challenges. In this area, the Apache Hadoop is the most renowned parallel framework. Not only used to achieve the high availability, Apache Hadoop is designed to detect and handle failures at the application as well as to maintain the data consistency.

Along with the development of Hadoop, the Hadoop Distributed File System (HDFS) has been introduced to provide the reliable, high-throughput access to parallel computing. Gradually, it becomes a suitable storage framework for parallel and distributed processing, especially for MapReduce-type solution, which was originally developed by Google to cope with the Big Data.

For improving the fault tolerance and reliability as well as providing the high availability and high performance, HDFS is initially equipped with a mechanism to replicate three copies of every data file from time to time. As time goes by, this replication strategy consumes storage resource and adds extra overhead to the system by making replicas of less frequently accessed data. Furthermore, although the speed of reading operation in HDFS might be improved by the available data, the performance of writing operation suffers the side effect of over-synchronizing unpopular data. Thus, it is reasonable to make an inference that the static replicating mechanism results the entire system in poorer performance than the benefit it contributes.

In this paper, a framework which makes the replicating component more effective is proposed. Not only is the nature of the accessing frequency is taken into account, but also the replica placement is considered. Subsequently, the data file in HDFS is replicated based on its utilization potential as well as the overall status of the system. Moreover, the anticipated result and access pattern are stored in the knowledge base for instantly matching and quickly firing the suitable action without re-calculating on similar input. From time to time, each data file would be efficiently replicated by differently appropriate strategy. By implementing this framework, the task execution time of Hadoop can be improved and benefit the productivity of Big Data system.

The remainder of this paper is organized as follows. In Section 2, related works relevant to the topic are provided. Section 3 presents an architectural overview of the background while Section 4 includes the detail of the prediction mechanism. In Section 5, the performance evaluation for the prediction framework is conducted. Conclusion is summarized in Section 6.

## II. Related Works

### A. System architecture

The Cost-effective Dynamic Replication Management (CDRM) [1] is a cost-effective framework for replication in cloud storage system. When the workload changes, CDRM calculates the popularity of 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 can not adapt well to the rapid evolution in large-scale systems.

Similarly, DARE [2] is another reactive model of replication for HDFS. In this model, the probabilistic sampling and competitive aging algorithm are used independently on each node to choose a replicating scheme for each data file as well as to decide the suitable location for each replica. Because DARE shares the same reactive-oriented approach like CDRM, it also shares the similar mal-adaptive properties.

978-1-4673-8537-4/15 $31.00 © 2015 IEEE
DOI 10.1109/CBD.2015.19

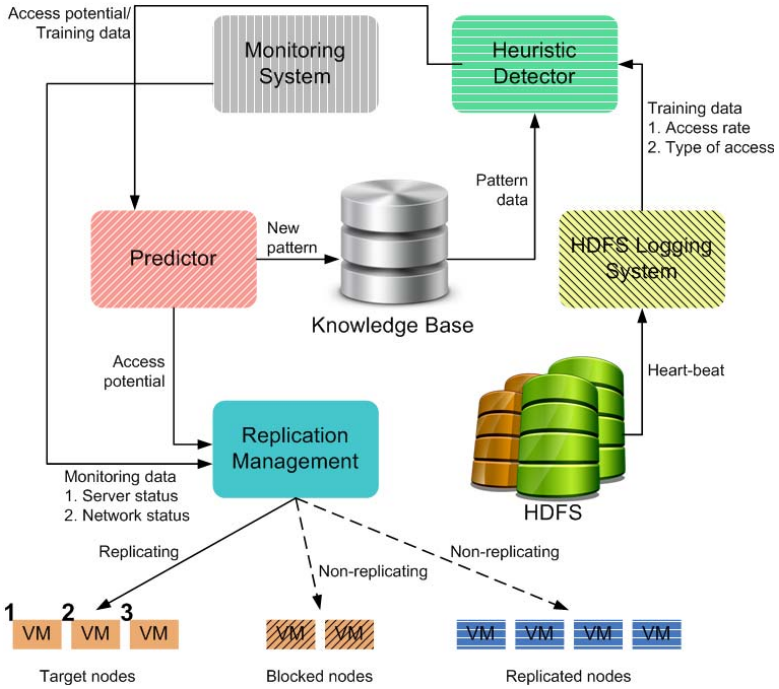



Fig. 1: Architecture of Adaptive Replication Management system (ARM).

The Elastic Replication Management System (ERMS) [3] takes into account an Active/Standby model for data storage in the HDFS cluster by implementing the complex event processing to classify the data types. The advanced concept of ERMS compared to CDRM and DARE is to dynamically change the thresholds for metrics based on the status of the HDFS cluster system. In addition, ERMS is equipped with the ability to determine and erase unpopular replicas to save the storage.

Nevertheless, although CDRM, DARE and ERMS are developed in different ways, all of them encounter the same problem and limitation. Concretely, these solutions try to classify and implement various replicating scenarios for each type of data file by extracting and processing the obsolete information from the monitoring statistic. For that reason, these approaches can not generate an optimal replication strategy for parallel systems. Clearly, the reason is when some actions are chosen to handle 'hot' data files, due to high latency and delay, these files may not be 'hot' anymore at the time the actions are engaged. Then, the replicating decision can not reflect the trend of data utilization.

The approach in Scarlett solution [4] implements the probability as an observation and then calculates the replicating scheme for each data file. Storage budget-limitation is also considered as a factor when distributing the replicas. Although this solution follows a proactive approach instead of using thresholds as in the reactive model, the intensity characteristic of the data file as well as the suitable placement for replicas are not discussed thoroughly.

Lastly, in OPTIMIS [5], an interesting solution for anticipating the data file status has been proposed. In this approach, the data file is also classified and engaged in one of few different replicating scenarios based on the algorithmic prediction of the demand for data file utilization. However, the Fourier Series Analysis algorithm, which is usually selected to analyze in signal processing, is chosen for prediction without a compelling proof of the efficacy. As a consequence, this inappropriate choice can result in poor prediction.

By examining related works, we conclude that although the research on replication strategies exists, not many researchers have thoroughly considered access frequency in the relationship with the big picture of system status. Furthermore, since Hadoop is no longer a standalone solution, but gradually a complex ecosystem. In that case, a more adaptive replicating mechanism must be developed to enhance the performance of Big Data systems, especially in the execution time and data throughput.

## III. Proposed Architecture

The purpose of the proposed idea is to design an Adaptive Replication Management system (hereinafter, ARM) for HDFS. In other words, the main function of this system is to dynamically scale the replication factor as well as to smartly schedule the placement for replicas based on the access potential of each data file. Additionally, to reduce the calculation time, knowledge base and heuristic technique are implemented as the extra components to detect similarities in the access pattern between the in-processing file and the predicted one. Two files with similar access behaviors are treated by the same replication strategy. However, because these techniques are standardized and widely used in various systems to accelerate the decision making procedure, discussing them is not in the scope of this paper.

Constructing on top of the Hadoop File System and Open Nebula Orchestrator, the proposed system takes responsibility to manage the replication over the virtualized HDFS instances. An overview of the proposed ARM is described in Figure 1. In this architecture, instead of using the physical servers for the computing nodes, the virtual machines are preferred because of the flexibility, elasticity, multi-tenancy and on-demand deployment features.

As described in Figure 1, the system starts by periodically collecting the heart-beat. After that, the training data goes to the Heuristic Detector as input. This training data is compared with the access pattern, which extracted from the Predictor component and stored at the knowledge base. The pattern is actually a set of eigenvectors describing the properties of the previously processed data and be used for matching purpose at the Heuristic Detector. At this stage, the features of input are compared with the pattern to look for similarities. If there is a matched, the access potential is retrieved from that pattern and directly passed to the Predictor component without any computation. Therefore, only one kind of data arrives at the Predictor, training data or the extracted access potential, but not both at the same time.

Before discussing the Predictor, it is essential to introduce the Monitoring System. Mainly based on the Ganglia framework, the Monitoring System is simple, robust and easy to configure for monitoring most of the required metrics. After plugging in the HDFS virtual nodes as well as the physical servers, the Monitoring System can collect statistic via Ganglia API. This information helps the replication management to determine which virtual node is busy and which physical server is blocked.

If the access potential comes to the Predictor as described in Figure 2, it means the training data is matched with a previous access pattern. Then, this access potential is directly forwarded to the Replication Management to create a replication strategy. No computation is needed. Otherwise, if the training data comes, the Predictor has to compute to achieve the access potential for files.

## IV. Prediction Model

The object of prediction in ARM is to anticipate the access potential of the data file. To obtain this target, Bayesian learning and Gaussian process are employed as the inference technique and probability framework, respectively. Assume that the input data is a time location set $x = [x_1, x_2, x_3, \cdots x_n]$, a finite set of random variables $Y = [Y_1, Y_2, Y_3, \cdots Y_n]$ represents the corresponding joint Gaussian distribution of access rate with regard to the time order. This set over the time constraint forms the Gaussian process.

$$f(y|x) \sim \mathcal{GP}\big(m(x), k(x, x')\big) \tag{1}$$

with

$$m(x) = \mathbb{E}\big(f(x)\big) \tag{2}$$

$$k(x, x') = \mathbb{E}\Big(\big(f(x) - m(x)\big)\big(f(x') - m(x')\big)\Big) \tag{3}$$

in which, $m(x)$ is the mean function evaluated at the location $x$ variable, $k(x, x')$ is the covariance function, also known as the kernel function [15]. Usually, the Square-Exponential (SE) kernel, also known as the Radial Basis Function (RBF) kernel, is chosen as follows.

$$k_{SE}(x, x') = \sigma_f^2 \exp\left(-\frac{(x - x')^2}{2l^2}\right) \tag{4}$$

in which, $\sigma_f$ is an output-scale amplitude and $l$ is a time-scale of the variable $x$ from one moment to the next. $l$ also stands for the bandwidth of the kernel and thereby the smoothness of the function in the model. Besides, $l$ also plays the role of judgment for Automatic Relevance Detection (ARD) to discard the irrelevant input.

On the next step, the posterior distribution of the Gaussian process is evaluated. Assume that the incoming value of the input data is $(x_*, y_*)$, the joint distribution of the training output is $y$, and the test output is $y_*$, as in the form below.

$$p\left(\begin{bmatrix} y \\ y_* \end{bmatrix}\right) = \mathcal{GP}\left(\begin{bmatrix} \mu(x) \\ \mu(x_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K}(x, x') & \mathbf{K}(x, x_*) \\ \mathbf{K}(x_*, x) & \mathbf{K}(x_*, x_*) \end{bmatrix}\right) \tag{5}$$

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

$$m_* = \mu(x_*) + \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1}(y - \mu(x)) \tag{6}$$

$$C_* = \mathbf{K}(x_*, x_*) - \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1}\mathbf{K}(x, x_*) \tag{7}$$

then

$$p(y_*) \sim \mathcal{GP}(m_*, C_*) \tag{8}$$

The best estimation for $y_*$ is the mean of this distribution.

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

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

$$var(y_*) = \mathbf{K}(x_*, x_*) - \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1}\mathbf{K}(x, x_*) \tag{10}$$

## V. Performance Evaluation

### A. Experiments

For the evaluation, three experiments are used to evaluate the performance of the proposed ARM. The first experiment is conducted on the Facebook cluster trace, namely the Statistical Workload Injector for MapReduce (SWIM) [35] [36]. Typically, this is the workload replay scripts to generate the real-life workloads from a Facebook production system. By sampling the historical MapReduce cluster traces, the SWIM provides an efficient method to measure the effectiveness of the solution, which intends to improve Big Data replication on the realistic data set. Two sets of synthesized day-long workloads, namely Facebook_trace_01 (*FB-2009_samples_24_times_1hr_0.tsv*) and Facebook_trace_02 (*FB-2010_samples_24_times_1hr_0.tsv*) are studied. Each set contains 24 historical traces sampled on a 600-machines cluster. The second experiment is the famous TeraSort stress test released by Yahoo!. This is a benchmark program written in MapReduce and included by default in Hadoop distribution. Basically, the TeraSort builds a sample key structure by selecting the subsets from the input before submitting the job and

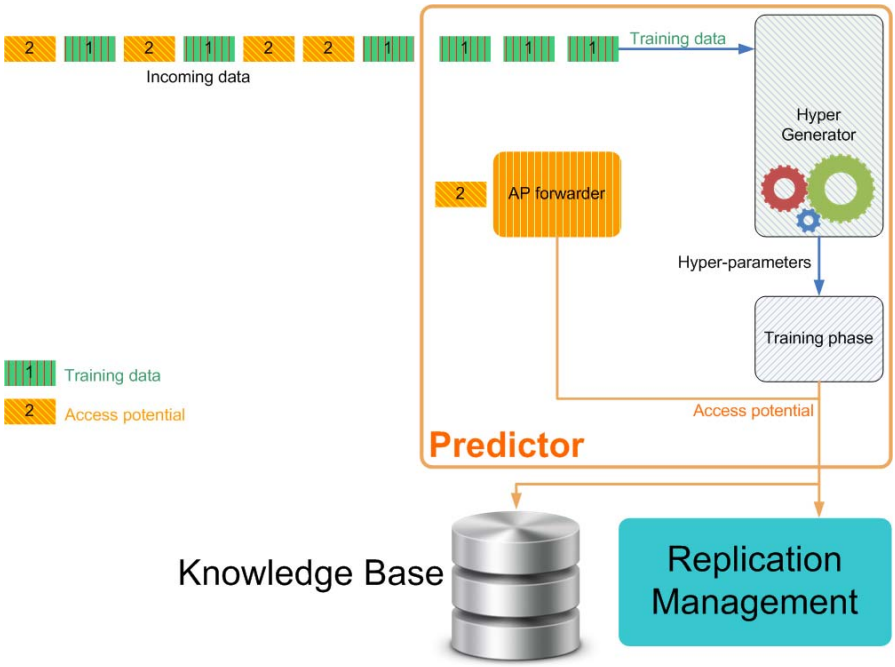



Fig. 2: Working mechanism of Predictor component.

TABLE I: System Configuration

| | Configuration |
|---|---|
| Computing Nodes | 01 Name Node, 8 Data Nodes |
| Node Types | XEN's Virtual Machine |
| Platform | 64bit |
| CPU Cores | Intel®Core™ i7-3770, 3.40GHz<br>4 cores for Name Node<br>1 cores for each Data Node |
| Storage | 500GB for Name Node<br>100GB for each Data Node |
| Memory | 16GB for Name Node<br>8GB for each Data Node |
| Network | Gigabit NIC |
| OS | CentOS 6.5 (final)<br>Kernel: 2.6.32-431.el6.x86_64 |
| Software | Apache Hadoop 2.0.0-cdh4.7.0 |

pushing this key structure into HDFS. Intuitively, the purpose of TeraSort is to sort a large volume of data rapidly. The third experiment is the TestDFSIO experiment. The TestDFSIO is a benchmark tool to discover the HDFS capability. In essence, this tool is developed to evaluate the I/O performance for HDFS. Additionally, the Hadoop version used in the evaluation is also modified to accept the dynamic replication factor as well as the flexible placement decision which is made by ARM system. The computing system used for these experiments is described in Table $I$.

### B. Metrics

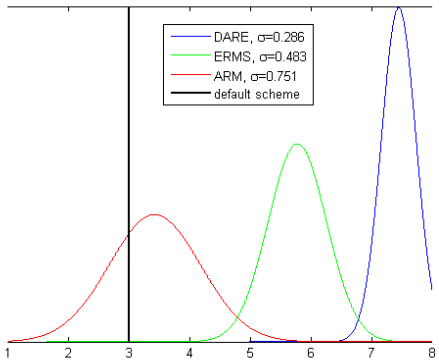
The data locality and data redundancy are the metrics of interest. In addition, some relevant factors such as read throughput and execution time are also considered. For the data locality evaluation, the metric is calculated in percentage as follows.

$$\mathcal{M}_{dl} = \frac{Access_{local}}{Access_{total}} \tag{11}$$

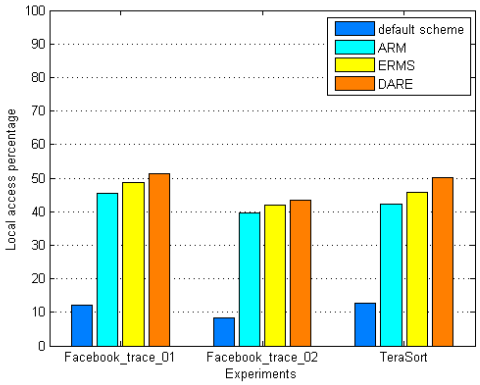
in which, $\mathcal{M}_{dl}$ is the data locality metric estimated by the fraction of the local access $Access_{local}$ over the total access $Access_{total}$ of every HDFS files. The counting for these accesses is executed automatically by the Monitoring System. Besides, the effect of data locality metric can be measured *via* the TestDFSIO experiments for execution time and read throughput of Hadoop system. The next system metric, as mentioned above, is the data redundancy. Obviously, if the HDFS replicating component creates too many replications, the data locality is improved in a spectacular rate. However, the side effect of this uncontrollable over-replication can rule the network to the congestion problem which directly makes the system unreliable and critically degrades the performance. Due to that reason, if the data locality can be improved with the well-bounded data redundancy, the performance of the whole Big Data system would be increased in terms of computational efficiency and network utilization. The data redundancy metric is evaluated *via* the shape of distribution (probability density function or pdf) of replication factors.

### C. Results

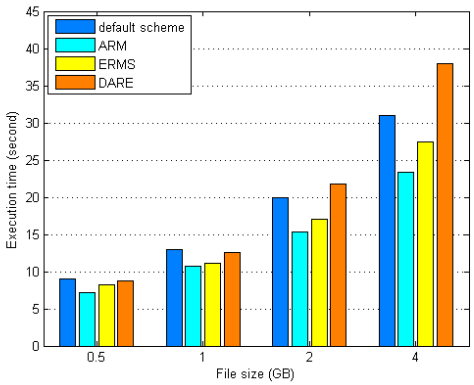
By averaging the results from both Facebook trace tests, ARM scores approximately 4 times better for the data locality



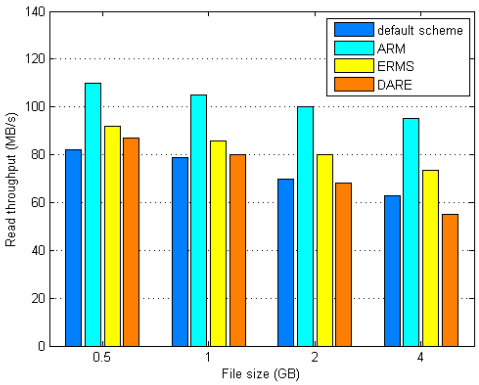

(a) Distribution (pdf) of each method.




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




(c) Execution time benchmark of TestDFSIO (lower is better).




(d) Read throughput benchmark of TestDFSIO (higher is better).

Fig. 3: Performance evaluation of proposed method on system level

metric in compared with the default scheme (Figure 3*b*). However, in these experiments, ARM gets lower result than ERMS and DARE (6.60% and 11,45% lower, respectively). In the TeraSort experiment (also in Figure 3*b*), ARM achieves 3.3 times improvement for the data locality compared with the default scheme. Clearly, the ERMS and DARE respectively continue scoring better results (8.25% and 18.53% higher) with regard to ARM. However, in Figure 3*a*, the variance shapes of replication factor distribution of ERMS and DARE are very narrow. It means these two methods usually set high replication factors for every data files to achieve better data locality metric. This behavior could definitely sacrifice the storage resource and network bandwidth.

On the execution time benchmark of TestDFSIO experiment (Figure 3*c*), it can be seen obviously that ARM outperforms ERMS and DARE when file size increases, even ARM has a lower score in previous data locality evaluation. In the last experiment case of 4GB of file size, ARM finishes the tasks in 17.38% and 62.36% faster than ERMS and DARE, respectively. The same situation happens when investigating the read throughput factor on Figure 3*d*. ARM continues surpassing ERMS and DARE in every experiment cases of file size. In fact, when engaging ERMS and DARE, the HDFS system seems to reserve too many CPU cycles for transferring and writing the replicas. It can lead the whole system into the disk-operation overhead. Subsequently, the transferring replicas may not be available to serve the computation.

## VI. Conclusion

The main purpose of this research is to improve the data locality metric by using the prediction technique. With rigorous analysis of the characteristics of file operation in HDFS, the uniqueness of our idea is to create an adaptive and effective solution to extend the capability of Big Data systems. For further development, some parts of the source code developed for testing our idea would be made available under the terms of the GNU general public license (GPL).

## Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) NRF-2014R1A2A2A01003914 and by the MSIP (Ministry of Science, ICT&Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014(H0301-14-1020)) supervised by the NIPA (National IT Industry Promotion Agency)

## REFERENCES


[1] 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.

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

[3] 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.

[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] J. Cunningham, Z. Ghahramani, and C. E. Rasmussen, "Gaussian processes for time-marked time-series data." in *AISTATS*, ser. JMLR Proceedings, N. D. Lawrence and M. Girolami, Eds., vol. 22. JMLR.org, 2012, pp. 255–263.

[7] R. Yates, *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers, 3rd Edition: Third Edition*, ser. Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers. Wiley Global Education, 2014. [Online]. Available: http://books.google.co.kr/books?id=zn5bAgAAQBAJ

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[9] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data." *CoRR*, vol. abs/1309.6835, 2013.

[10] G. Chowdhary, H. Kingravi, J. How, and P. Vela, "Bayesian nonparametric adaptive control using gaussian processes." *Neural Networks and Learning Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[11] R. Grande, G. Chowdhary, and J. How, "Nonparametric adaptive control using gaussian processes with online hyperparameter estimation." in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, Dec 2013, pp. 861–867.

[12] C. E. Rasmussen, "Evaluation of gaussian processes and other methods for non-linear regression." Ph.D. dissertation, Toronto, Ont., Canada, Canada, 1997, aAINQ28300.

[13] 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

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

[15] 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.

[16] 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.

[17] 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.

[18] E. Rodner, A. Freytag, P. Bodesheim, and J. Denzler, "Large-scale gaussian process classification with flexible adaptive histogram kernels." in *ECCV (4)*, ser. Lecture Notes in Computer Science, A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., vol. 7575. Springer, 2012, pp. 85–98.

[19] 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/

[20] 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.

[21] 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.

[22] 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

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

[24] D. Petelin, B. Filipič, and J. Kocijan, "Optimization of gaussian process models with evolutionary algorithms." in *Proceedings of the 10th International Conference on Adaptive and Natural Computing Algorithms - Volume Part I*, ser. ICANNGA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 420–429. [Online]. Available: http://dl.acm.org/citation.cfm?id=1997052.1997098

[25] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain." Pittsburgh, PA, USA, Tech. Rep., 1994.

[26] 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.

[27] C. Yang, R. Duraiswami, and L. S. Davis, "Efficient kernel machines using the improved fast gauss transform." in *NIPS*, 2004.

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

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

[30] 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

[31] R. S. Sampath, H. Sundar, and S. K. Veerapaneni, "Parallel fast gauss transform." in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/SC.2010.39

[32] V. Simoncini and D. B. Szyld, "Theory of inexact krylov subspace methods and applications to scientific computing." *SIAM Journal on Scientific Computing*, vol. 25, no. 2, pp. 454–477, 2003.

[33] V. I. Morariu, B. V. Srinivasan, V. C. Raykar, R. Duraiswami, and L. S. Davis, "Automatic online tuning for fast gaussian summation." in *NIPS*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2008, pp. 1113–1120.

[34] 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.

[35] 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.

[36] Y. Chen, S. Alspaugh, and R. H. Katz, "Interactive query processing in big data systems: A cross-industry study of mapreduce workloads." in *VLDB*. IEEE, 2012, pp. 390–399.

[37] T. Andrews, "Computation time comparison between matlab and c++ using launch windows." San Luis Obispo, SLO, CA, 93407, USA, Tech. Rep., 2012.