# FRLDM: A Self-Optimizing Algorithm for Data Migration in Distributed Storage Systems

Tao Wang[1, 2, a], Shihong Yao[1, b], Zhengquan Xu[1, 2, c, *], Shan Jia[1, 2, d], Lizhi Xiong[1, 2, e]

[1]State Key Laboratory of Information Engineering in Surveying Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China

[2]Collaborative Innovation Center for Geospatial Technology, Wuhan 430079, China

[a]wangtao.mac@whu.edu.cn, [b]yao_shi_hong@whu.edu.cn, [c]xuzq@whu.edu.cn, [d]jias@whu.edu.cn, [e]xionglz@whu.edu.cn

*Corresponding Author: xuzq@whu.edu.cn

**Keywords:** data migration, fuzzy approximation, reinforcement learning, self-optimize.

**Abstract.** In distributed storage systems, data migration is an efficient method for improving system resource utility and service capacity, and balancing the load. However, the user accessing is changing over time and the state of a distributed system is in an unpredictable stochastic fluctuation, hence traditional heuristic policy- based methods are hard to work in such environment. This paper proposes a fuzzy reinforcement learning method for online data migration named FRLDM which can enable the systems to self-optimize and dynamically choose the candidate data for migration based on their recent access pattern and the current state of the system, thus minimizing the average access response time. The experimental results prove that FRLDM can improve the accesses performance significantly compared with heuristic policy-based methods.

## 1. Introduction

In distributed storage systems, data migration is an efficient method for improving system resource utility and service capacity and balancing the load. Currently most researches on data migration are heuristic policy-based methods [1, 2], which heuristically specify which data, when and where to migrate at some certain conditions. They are easy to realize and can optimize the system performance temporally. However, data access patterns are more diverse and change over time [3], and the state of a distributed system is in an unpredictable stochastic fluctuation, such heuristic policy-based methods are hard to work in such environment and cannot adapt the changes in real-time, which may deteriorate the system performance in the long term. Additionally, in current enterprise-class storage systems, online data migration is required because of equipment damage and expansion [4]. Hence, to continually optimize the system performance, it is expected that the systems can develop and implement real-time migration decisions according to the system states and environment evolution. In other words, the storage systems should be intelligent which have the capacity of decision-making and self-optimizing without unfaithful human interventions.

Reinforcement learning (RL) [5] is a machine learning method which use experience gained through interacting with the world and evaluative feedback to continuously improve a system's ability to make behavioral decisions. Different with supervised learning and unsupervised learning, RL provides a high performance self-learning framework for the learning systems, where a key issue in RL is to construct the reward functions [6]. Fuzzy logic approximator (FLA) [7] is an efficient and widely-used method for function approximation which can approximate the reward functions in RL. Then, the self-learning capacity of RL and the approximation capability of FLA are integrated to provide a novel solution for realizing the self-optimization in large-scale distributed storage systems.

This paper proposes a data migration method based on multi-agent fuzzy RL, which establishes the optimization goal for distributed storage systems, then uses FLA to approximate the reward

 

functions and RL to learn and adjust the parameters. Accordingly, the automatic online data migration is realized to improve access performance under continuous and large-scale state space and stochastic dynamic environment.

## 2. Data Migration

A Large-scale storage system is composed of data centers (in cloud computing) or clusters (in data grids) [8], where any data center or cluster is denoted as $C_i$ and its shared bandwidth for data I/O and data migration is $B_i$. Every single file in the system has multiple replicas, and the I/O request for each file is stochastic and scheduled to the data center which has the smallest response time. The response time of an I/O request consists the queuing time and the processing time, and the processing time is determined by the size of requested files divided by shared bandwidth of the local data center.

Some files are selected and migrated from their located data center to another to improve their current response time, and meanwhile, the expected future response time of the other files in the original data center and destination data center will be affected. Hence, the influence on the future states of the involved data centers must be considered in data migration process, which is neglected in policy-based methods.

Reward functions are usually adopted to represent expected optimizing objectives. Once a reward function is formed, the system will take actions automatically to optimize current reward functions. A reward function reflects not merely the current system state but also the future state after a sequence of actions. Therefore, how to construct the reward function for a distributed system is the necessary prerequisite for realizing the RL and the self-optimized data migration.

## 3. Fuzzy Logic Approximator

Fuzzy logic approximators (FLAs) have been successfully applied in functional approximation, modeling and control in dynamical systems due to the approximation capabilities and inherent adaptive features, and proven to be capable of approximating any well-defined nonlinear function to any degree of accuracy [9].

The fuzzification maps some deterministic inputs into some fuzzy sets and uses membership functions determine the degree to which an input belongs to a fuzzy set. The fuzzy If–Then rules are:

$$Rule^{(l)}: \text{IF } x_1 \text{ is } A_1^l \text{ and } \cdots \text{and } x_N \text{ is } A_N^l, \text{ THEN } y^l \text{ is } B^l \quad . \tag{1}$$

Where $x_n$ ($n \in [1, N]$) are the inputs of the system, $y^l$ is the output, $A^l{}_n$ and $B^l$ ($l \in [1, L]$) are the fuzzy set under rule $l$, and $L$ is the number of fuzzy rules. In general, $B^l$ is a unique point. Membership function $\mu_{A_n^l}(x_n)$ which is of the Gaussian form determines the degree to which the $x_n$ belongs to the fuzzy set $A^l{}_n$. The defuzzification translates the fuzzy output into the deterministic output:

$$f(x) = \sum_{l=1}^{L} \frac{y^l \prod_{n=1}^{N} \mu_{A_n^l}(x_n)}{\sum_{l=1}^{L} \left( \prod_{n=1}^{N} \mu_{A_n^l}(x_n) \right)} . \tag{2}$$

Where $x_n \in [0,1]$, $x = (x_1, x_2, \cdots, x_N)^{\mathrm{T}}$, and the activation degree of rule $l$ is $\omega^l(x) = \prod_{n=1}^{N} \mu_{A_n^l}(x_n)$. There are some overlaps among the fuzzy sets which have no obvious boundaries, and $y^l$ changes continuously with the extent to which $x$ belongs to the fuzzy sets. That can avoid the sudden change of $y^l$ when $x$ crosses the boundary between two input spaces in traditional heuristic policy-based methods. As a result, FLA is very popular in parameter optimization of complex control systems.

## 4. The Method of Fuzzy Reinforcement Learning

A data center can be viewed as a learning system (LS). It is infeasible to define a global reward function because of the dynamic add and quit of a LS. Every learning system needs a reward function $R(s)$ of its own. $R(s)$ indicates the expected future response time of the data center at the current state

$s$. The more accurately the reward function is learned, the more benefits the data migration will bring.

**4.1 Reward Functions.**

Appropriate input parameters are selected by the system for each reward function and the outputs are translated into the data migration strategies. For distributed storage systems, the learning time of a reward function grows exponentially with the number of dimensions of state variables [10], so state variables are preferred to be fewer and can reflect the future system state.

A two-dimensional state vector $s=(s', s'')$ is used to represent a data center's state. $s'$ indicates the average access rate of all files in a data center. When a file *F* is accessed at time *t*, its average access interval is $\tau_F = \mu\tau_F + (1-\eta)(t-t')$, where $t'$ is the last access time of the file *F*, and $\eta<1$ is the discounted coefficient which determines the discounted rate of history access information. Then, the access rate of file *F* is approximated as $a_F = 1/\tau_F$. $s'$ can reflect the number of accesses per unit time that the data center will receive in the future, and the larger $s'$ is, the smaller $R(s)$ will be. $s''$ is the current queuing time in the data center, and a larger $s''$ suggests a longer expected access response time in the future. When the current states of data centers $dc_i$ and $dc_h$ are $s_i$ and $s_h$, and the current rewards are $R_i$ and $R_h$, respectively, the file will be migrated between the data centers $dc_i$ and $dc_h$ if and only if $(R_i s_i + R_h s_h) > (\tilde{R}_i \tilde{s}_i + \tilde{R}_h \tilde{s}_h)$, where $\tilde{s}_i$ and $\tilde{s}_h$ are the resulting states of data centers $dc_i$ and $dc_h$, and $\tilde{R}_i$ and $\tilde{R}_h$ are the resulting rewards after migration. This method is robust to the dynamic add and quit of a data center and scalable.

**4.2 Fuzzy approximation of reward functions.**

The numbers of fuzzy sets and fuzzy rules heavily influence the complexity of a FLA, and the larger the numbers are, the higher the approximation accuracy will be, and the more complex FLA will be. When there are sufficient fuzzy sets and appropriate membership functions, FLA can approximate any continuous function [7]. However, the trade-off between complexity and accuracy must be considered in real-world usage.

As shown in Fig. 1, three membership functions are employed to describe the degree to which the input $x_n$ belongs to the three fuzzy sets, COLD, WARM and HOT when $x_n$ takes value in the interval [0, 1]. Whenever a data migration is needed, each data center will calculate its $R(s)$ and $R(\tilde{s})$ at the input of $x=(s', s'')$ or $x=(\tilde{s}', \tilde{s}'')$ and then the output $y^l$ is resulted by Eq.(2).

**4.3 FRLDM: FRL-based Online Data Migration Method.**

A distributed storage system has a large-scale and continuous state space, and data migration can be triggered at any time point, so the migration decision process of a LS can be viewed as a Semi-Markov decision process (SMDP) and a Partially Observable Markov Decision Process (POMDP) because it is hard to acquire all relevant information for describing the system states and constructing reward functions. A memoryless method [11] is used in this paper, and only the observed state vector of each LS is used to learn the reward function of each LS.

The I/O-triggering is used in the data migration strategy in this paper. To learn the reward functions more accurately, a LS will not update its reward function until at least *P* I/O requests have been processed since the last migration decision. Then, the data migration is triggered at irregular time intervals because the time costs of processing *P* I/O requests are different for different LSs.

Given a set of states *S* and a set of actions *A*, a LS takes an action $a_m \in A$ in the state $s_m \in S$ at time $t_m$, and then transfers into the state $s_{m+1} \in S$ at time $t_{m+1}$ with the state transition probability $P(s_{m+1}|s_m, a_m)$. The expected reward of a LS under a policy $\pi$ at state $s$ is

$$V^{\pi}(s) = E\left[\sum_{m=0}^{\infty}\int_{t_m}^{t_{m+1}} e^{-\alpha t} r(s_m, s_{m+1})dt \,|\, s_0 = s, \pi\right], \tag{3}$$

Where $\alpha$ is the discounted coefficient.

As known that RL is usually applied to solve the problem of delayed reward, however, each migration action would influence the states of the involved data centers immediately in a distributed storage system. Temporal difference (TD) learning is an efficient RL algorithm which uses the

difference value between two consecutive time points to update the reward functions. For a SMDP with finite state and state transitions in a finite period of time, TD learning has a good performance on convergence and availability [12]. Hence, TD learning is employed in this paper.

Suppose the difference value at time $t_m$ is:

$$\Delta_m = r_m + e^{-\alpha\tau_m} V^{\pi}(s_{m+1}) - V^{\pi}(s_m), \tag{4}$$

Where $s_m$ and $s_{m+1}$ are the states at time $t_m$ and $t_{m+1}$, $\tau_m$ is the time the LS stay in state $s_m$, $r_m$ is the accumulated reward of the LS in state $s_m$. To accelerate the learning and realize incremental learning, the eligibility trace function [13] of each state is introduced:

$$z_m(s) = \begin{cases} \lambda e^{-\alpha\tau_m} z_{m-1}(s), & s \neq s_m \\ \lambda e^{-\alpha\tau_m} z_{m-1}(s) + 1, & s = s_m \end{cases}, \tag{5}$$

Where $\lambda$ is a constant and $0 \le \lambda \le 1$. This function records the recent occurrence frequency of each state in a circular manner, and when a state occurs, its eligibility trace will increase, otherwise its eligibility trace will decrease exponentially. This function determines how relevant the states and what have just happened are. When $\lambda = 0$, only the value function of the last state occurred is updated. Then the value function is:

$$V^{\pi}(s) = V^{\pi}(s) + \rho_m \Delta_m z_m(s), \tag{6}$$

Where $\rho_m$ is the learning coefficient at time $t_m$. When a state transition occurs, the value function iteration is executed simultaneously for all $s \in S$. However, it is infeasible to assign a reward function to each state due to the state space of a distributed storage system is very large and continuous, that will lead to a low learning efficiency of TD and high costs of compute and storage. Therefore, a non-linear value function approximation based on the neighboring states is adopted to approximate the optimal value function $\hat{V}^{\pi}(s)$:

$$\hat{V}_m(s_m) = \phi^T(s_m) Y_m = \sum_{l=1}^{L} \phi^l(s_m) y^l, \tag{7}$$

Where $\phi(s) = \left(\phi^1(s), \phi^2(s), \cdots, \phi^L(s)\right)^T$ the basis function vector of state s is, $\phi^l(s)$ is linearly independent which the normalized weight of rule l is:

$$\phi^l(s_m) = \frac{\omega^l(s_m)}{\sum_{l=1}^{L} \omega^l(s_m)}. \tag{8}$$

Then, when the current state is $s_m$, the iterative rule of fuzzy output is

$$y^l_{m+1} = y^l_m + \rho_m \left[ r_m + e^{-\alpha\tau_m} \hat{V}_m(s_{m+1}) - \hat{V}_m(s_m) \right] z^l_{m+1}, \tag{9}$$

Where $\rho_m = 1/m$ is the learning coefficient, and $r_m$ is the average weighted response time of all requests processed by the data center between the state $s_m$ and the state $s_{m+1}$:

$$r_m = \frac{1}{U_m} \sum_{u=1}^{U_m} \delta_u e^{-\alpha\left(t^u_m - t_m\right)}, \tag{10}$$

Where $U_m$ is the number of requests processed in state $s_m$, $\delta_u$ and $t^u_m$ are the response time and the arrival time of the *u*th request in state $s_m$. $z^l_{m+1}$ is initialized as $z^l_0 = 0$, and its iterative rule is :

$$z^l_{m+1} = \lambda e^{-\alpha\tau_m} z^l_m + \frac{\partial \hat{V}(s_m)}{\partial y^l_m} = \lambda e^{-\alpha\tau_m} z^l_m + \phi^l(s_m). \tag{11}$$

When the result of value function stops changing observably in the iteration of TD, the value function is considered to be an approximation to the optimal $\hat{V}^{\pi}(s)$ and parameters of the value function begin to be convergent. For data migration, each file migration would influence the current state of the data centers immediately, so the policy iteration only needs to consider the current value functions of each state, and the optimal policy $\pi'$ can be obtained by the following iterative rule:

$$\pi'(s_m) = \arg\max_{a \in A} E\left\{V^{\pi}(s_m)\right\} \tag{12}$$

Where $r(s_m, a)$ is the reward gained in state $s_m$ after taking the action $a$. When the policy $\pi$ has a positive impact on the value function, the LS will greedily implement the policy based on the current value function and not change the policy until the value function reaches convergence.

FRLDM algorithm is shown in Table 1.

Table 1 FRLDM algorithm

**Step 1:**Initialize $t_m = 0$, $m = 0$, the sate $s_m = (s'_m, s''_m)$ at time $t_m$ of each LS; compute the reward by **(10)**, and construct fuzzy sets: HOT、WARM and COLD and fuzzy rules and membership functions. In addition, set the learning coefficient as $\rho_m = 1/M$.

**Step 2:**Initialize value function $V^{\pi}(s)$ and the eligibility trace function as $z_m(s) = 0$.

**Step 3:**Approximate the optimal value function $\hat{V}^{\pi}(s)$ and loop:

**Substep 1**: compute the difference at time $t_m$ by **(4)** and the eligibility trace $z_m(s)$ of the state *s* by **(5)**

**Substep 2:** compute the basis function vector $\phi^l(s)$ of state *s* by **(8)**

**Substep 3:** update the value function estimation $V^{\pi}(s)$ of state *s* by **(6)**

**Substep 4:** update the eligibility trace function $z^l$ by **(11)**,the average weighted response time $r_m$ by **(12)** and the fuzzy output $y^l$ by **(9)**

**Substep 5:**update the value function till it approximates the optimal value function $\hat{V}^{\pi}(s)$ by **(11)**

**Substep 6:** $m = m+1$ and return to **Substep 1**

**Step 4:** compute the optimal policy $\pi'$ by **(12)**

**Step 5:** trigger the data migration when the criterion is satisfied and quit, otherwise, return to **Step 2**

The procedure of policy improvement is also known as policy iteration (PI) and it can find a good policy which only requires several iterations and never adopts a bad policy [14]. Hence, PI has attracted extensive attention in online learning. The distributed storage system implements the fuzzy reinforcement learning in the manner of multi-agent to realize the dynamic data migration decision and the system self-optimizing.

## 5. Simulations and Results Analysis

The performance of data migration method is evaluated by the average response time. Due to the high complexity of FRLDM implementation, some assumptions are simplified to focus on the availability of FRLDM and performance evaluation under different data migration strategies. CApp [15] is used as the simulation environment. Some key parameter configurations are shown in Table 2.

Table 2 Parameters configuration

| | |
|---|---|
| The number of files | 5000 |
| The number of replica of each file | 1 |
| The state categories of a file | HOT,WARM, COLD |
| Initial probability of a file being HOT, WARM or COLD | 1/3 |
| The state transition probability of a file been accessed | P(WARM｜COLD)=0.3,<br>P(HOT｜COLD)=0.1<br>P(HOT｜WARM)=0.3<br>P(COLD｜WARM)=0.1<br>P(WAR｜MHOT)=0.01<br>P(COLD｜HOT)=0.005 |
| Access distribution of files | Poisson distribution |
| Arrival rate of file access | HOT:0.5<br>WARM:0.1<br>COLD:0.01 |

Whenever any data center has processed at least *K* I/O requests since the last data migration, $y_m^l$ is initialized to 0 and executed simultaneously for all data center. The set value of *K* is important which ranging from 40 and 90 has nearly the same performance experimentally. In addition, the value of $e^{-\alpha}$ which is between 0.9 and 0.95 has the approximate optimal performance experimentally. The algorithm consists of the training stage that updates parameter 5000 times and the testing stage that evaluates the different data migration policies during another 5000 time steps.

Three membership functions shown in Fig.1 are used to approximate reward functions. To select the appropriate values for mean $\bar{x}_n$ and the standard deviation $\sigma_n$, the first 1000 data migration decisions are observed to calculate them of each input parameter.

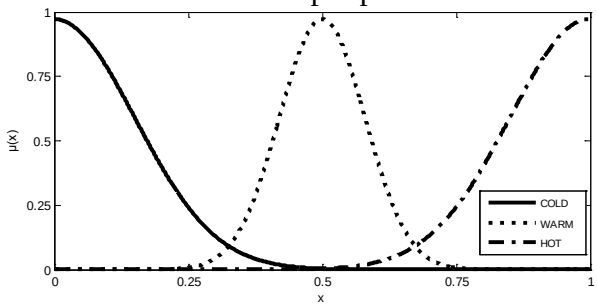


Fig.1 The Membership Functions of The Three Fuzzy Sets: COLD, WARM and HOT

It is obvious shown in Table 3 that there are no obvious statistical variations of MSEs when the $\lambda$ takes the different value. In other words, the FRLDM algorithm is robust to the various values of $\lambda$.

Table 3 The MSEs of FRLDM at different λ

| $\lambda$ | 0 | 0.5 | 1 |
|---|---|---|---|
| MSEs | 0.039 | 0.057 | 0.062 |

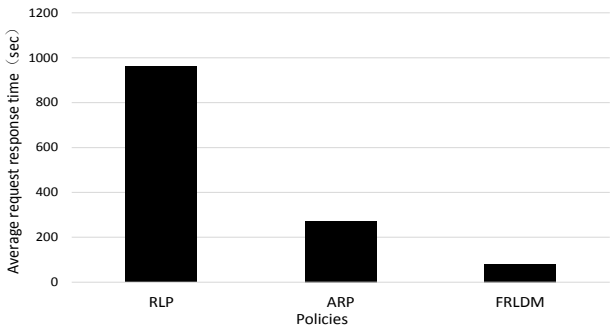


Fig.2 The average response time under different strategies

Fig.2 shows the average response time of different policies and the performance of FRLDM is superior to RLP and ARP significantly. The first policy is the random layout policy (RLP) which allocates all files into data centers randomly and doesn't consider data migration. The second policy is the access rate policy (ARP) which migrates the files with the highest access rate in the data center that has the highest $s'$ to the data center that has the lowest access rate $s'$. ARP is also a load balancing policy. The last policy is the FRLDM which maximizes the value function.

## 6. Conclusions

Data migration is an efficient method for data redistribution. This paper proposes a fuzzy reinforcement learning method for online data migration in distributed storage systems named FRLDM which integrates the self-learning capacity of RL and the approximation capability of FLA to find the optimal data migration policy. FRLDM can enable the systems to self-optimize and dynamical data migration. The results prove that FRLDM is robust and can reduce the average response time compared with heuristic data migration policies.

This paper mainly focuses on the availability of the proposed FRLDM. However, there are some implementation details which are not considered because of the complexity of the distributed storage

systems, and in the future works, the more complex system states and the more specific implementation details will be studied.

**Acknowledgements**

The research work reported in this paper is supported by the National Natural Science Foundation of China (No. 41371402 and No. 41271398), the National Basic Research Program of China (No. 2011CB302306) and the National Natural Science Foundation of China under Grant (No: 61402421).

**References**

[1] B. Dong, X. Li, L. Xiao, L. Ruan. Self-acting load balancing with parallel sub file migration for parallel file system. 3rd International Joint Conference on Computational Science and Optimization. Huangshan, Anhui, 2010, p. 317–321.
[2] Tan Z, Zhou W, Feng D, et al. ALDM: Adaptive Loading Data Migration in Distributed File Systems. IEEE Transactions on Magnetics. Vol. 49(2013) No. 6, p. 2645 - 2652.
[3] Kang S, Reddy A L N. User-Centric Data Migration in Networked Storage Systems. IEEE International Symposium on Parallel and Distributed Processing. Miami, FL, USA, 2008, p. 1-12.
[4] Gong Zhang, Lawrence Chiu, Ling Liu. Adaptive Data Migration in Multi-tiered Storage Based Cloud Environment. The 3rd IEEE International Conference on Cloud Computing. Miami, FL, USA, 2010, p. 148-155.
[5] Barrett E, Howley E, Duggan J. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. Concurrency & Computation Practice & Experience. Vol. 25(2013) No. 12, p. 1656–1674.
[6] Wu J, Xu X, et al. A novel multi-agent reinforcement learning approach for job scheduling in grid computing. Future Generation Computer Systems. Vol.27 (2011) No.5, p. 430–439.
[7] Zhi Liu, Han Xiong Li. A probabilistic fuzzy logic system for modelling and control. IEEE Transactions on Fuzzy Systems. Vol. 13 (2005) No. 6, p. 848-859.
I. Foster, Z. Yong, I. Raicu, S. Lu. Cloud computing and grid computing 360-degree compared. Grid Computing Environments Workshop. Austin, TX, USA, 2008, p. 1–10.
[8] Zhou Q, Shi P, Xu S, et al. Adaptive output feedback control for nonlinear time-delay systems by fuzzy approximation approach. IEEE Transactions on Fuzzy Systems. Vol. 21(2013) No. 2, p. 301-313.
[9] Duell S, Udluft S, Sterzing V. Solving Partially Observable Reinforcement Learning Problems with Recurrent Neural Networks. Lecture Notes in Computer Science, 2012, 7700, p. 709-733.
[10] Li Y, Yin B, Xi H. Finding optimal memoryless policies of POMDPs under the expected average reward criterion. European Journal of Operational Research. Vol. 211(2011) No. 3, p. 556-567.
[11] Littman M L. Reinforcement learning improves behaviour from evaluative feedback [J]. Nature. Vol. 521 (2015) No.7553: 445-451.
[12] Främling K. Replacing eligibility trace for action-value learning with function approximation. Proceedings of 15th European Symposium on Artificial Neural Networks. Bruges, Belgium, 2007, p. 313-318.
[13] Bertsekas D P. Approximate policy iteration: A survey and some new methods [J]. Journal of Control Theory and Applications. Vol. 9 (2011) No. 3: 310-335.
[14] Tao Wang, Shihong Yao, Zhengquan Xu, et al. DCCP: an effective data placement strategy for data-intensive computations in distributed cloud computing systems [J]. Journal of Supercomputing, (2015), DOI: 10. 1007/s11227-015-1511-z