Proactive and Adaptive Data Migration in Hierarchical Storage Systems using Reinforcement Learning Agent

T.G.Lakshmi
Department of Computer Engineering
Thakur College of Engineering and Technology
Mumbai, India - 400101

R.R.Sedamkar
Department of Computer Engineering,
Thakur College of Engineering and Technology
Mumbai, India - 400101

ABSTRACT
With the data generation rates growing exponentially, businesses are having a difficult time maintaining data center infrastructure. Hierarchical storage systems has evolved as a better alternate to managing data, as frequently accessed data is placed on higher tiers and the least frequently accessed data on lower tiers. But the data arrangement is not always static. Data Migration is an operation in which the selected data is physically moved, or migrated, to different storage components. The existing method of data migration in hierarchical storage systems has several shortcomings – reactive, heuristic policies and proprietary software. This paper attempts to overcome the above limitations by applying a reinforcement learning (RL) agent in the data migration of hierarchical storage systems. By the addition of RL agent the data migration which earlier was responsive, is made proactive and adaptive. Experimental results demonstrate the effectiveness of the proposed RL agent. The results indicate that: (i) the average queue size of storage devices is reduced to almost zero as the RL agent proactively migrates data and (ii) at 95 % confidence level the RL agent has no affect on the read and write operation of certain file sizes.

General Terms
Data Migration, Hierarchical Storage Management

Keywords
HSS, Multi-Tier storage, Data Management, Data Migration, AI, Reinforcement Learning, Q-Learning

1. INTRODUCTION
Storage of information is the most vital function in an organization. A large quantity of digital information is being created every moment by individual and corporate businesses [1].Storage area networks are now recognized as the preferred solution for fulfilling a wide range of critical data storage needs for institutions and enterprises [2]. But in recent times the data is exploding and to keep pace with it the primary and backup infrastructure is growing massively. There seems to be one more problem that accompanies the data and infrastructure growth – data utilization. Even though the data and infrastructure are growing exponentially the utilization of data does not increase at the same rate. It becomes more difficult to determine whether such data can be erased or deleted permanently due to compliance policies such as PCI DSS [3].

The cost of the data center operations is becoming difficult to contain. The problem is not only with the data growth which leads to the acquisition of new disk arrays but it gets compounded by the (1) backup,(2)floor space and (3) electricity costs. Hierarchical Storage Management (HSM) provides an automatic way of managing and distributing data between the different storage layers to meet user needs for accessing data while minimizing the overall cost [4]. Hierarchical Storage Management (HSM) is used to effectively utilize the storage space according to its capability and also meet customer demands [4]. In HSM the storage devices are categorized into tiers based on their access speed. TIER-I storage would consists of devices which have higher I/O throughput and hence expensive. The most frequently used data is placed on TIER-I. The data arrangement is not static as the value of data changes through its lifetime. The purpose of HSM would be defeated if data once placed on a storage tier remains there till the end. Data needs to be relocated or migrated as and when its value decreases or increases.

Data Migration is the term given to the action of displacing data from one storage tier to another storage tier. In a typical SAN, this action can be initiated either from the storage controller end or host end as seen from figure 1. Data migration when implemented at the storage controller tends to become proprietary and hence would not work on heterogeneous systems. At the host end data migration techniques have been reactive and rule-based policies. A typical example of a rule based policy - IF space utilization in TIER-I is greater than 80%, THEN migrate *.tmp files greater than 1MB ordered by size to TIER-III until the space utilization of TIER-I is 60% [4].

To overcome the above flaws in data migration, in this paper a Reinforcement Learning (RL) agent to perform data migration in hierarchical storage systems is used. The RL agent performs anticipatory data migration based on the device's characteristics – (i) transfers per second, (ii) read/write per second and (iii) average queue size. The RL agent keeps observing the device characteristics and as it sees the value deteriorate it schedules for data migration appropriately. The data migration rules are learnt over a period of time based on positive and negative rewards. To evaluate the effectiveness of the RL agent the following study was conducted. With the same Input/output (I/O) traffic we compared the average queue size of 3 separate devices. After around 500 observations, the RL agent brought down the queue size to almost zero in all the three devices. The write and read performance of devices with and without RL agent was compared using t-tests. For certain file sizes the t-test evaluation supports the premise at 95 % confidence levels, that the RL agent does not affect the I/O performance of the device.

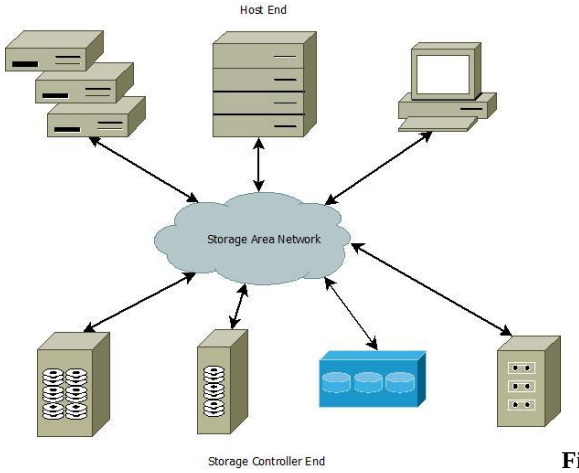

Fig 1: Typical Storage Area Network

The next section is organized as follows: Section 2 introduces the related work and also provides a background on the application of reinforcement learning. In Section 3 and 4, we describe the research questions and details of the implementation. In Section 5 we present the experiment settings and results thereof. We conclude with a discussion in Section 6.

2. RELATED WORK

Goal oriented learning involves constant interaction with the environment. The environment interface gives a lot of information about the cause – effect relationship leading to the knowledge about the consequences of action. The knowledge gained, helps in determining the actions required to achieve the goal. Animal learning, including human infants is based on the above approach. Reinforcement Learning (RL) is a goal- directed computational approach to learning from interaction [5]. Work by A. Harry Klopf (late 1979) on "Heterostatic theory of adaptive systems" became the base for future work in RL [5]. RL dates back to the early days of cybernetics and work in statistics, psychology, neuroscience, and computer science [6]. RL is one of the most active research areas in machine learning and artificial intelligence.

Reinforcement Learning does not need prior knowledge; it autonomously gets optimal policy with the knowledge obtained from trial and error and continuous interaction with the dynamic environment. It has characteristics of self-improvement and online learning. In RL the learner is learning how to map situations to actions so as to maximize the long term reward [6]. The learner is not told the best action for a situation but discovers the best action for a situation by trial and error [6]. There are several differences between RL and standard supervised learning. In RL correct input/output pairs are never presented and there is no correction mechanism for sub-optimal actions. RL has an additional focus towards striking the balance between exploration and exploitation.

Data Migration closely resembles a 'Markov Decision Process' (MDP). A MDP is a discrete time stochastic control process [7]. At each time step, the process is in some state S, and the decision maker/agent may choose any action 'a' that is available in state 'S'. The process responds at the next time step by randomly moving into a new state S', and giving the agent a corresponding reward Ra (S, S'). MDP can be solved via Dynamic Programming and Reinforcement Learning [7]. Dynamic Programming has several shortcomings: (1) knowledge of the complete environment is unobtainable; (2) number of states becomes computationally intractable as the problem size increases. The Reinforcement learning model is presented in figure 2.

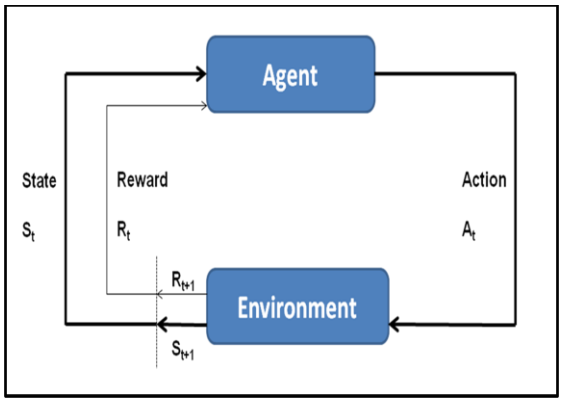

Fig 2: Reinforcement Learning Model

2.1 Data Storage Management using Artificial Intelligence Methods

AI techniques have been exploited in the proactive management of computer networks. The study investigated the use of 2 artificial intelligence methods fuzzy logic and rules for data location and replica management in distributed data storage [8]. The proposed system helped select the cheapest location for file creation and download. It also performed replica management such as creation and deletion of replica. The study was performed with the help of 43 events and it was observed that the system was able to adapt to the observed situation in the monitored infrastructure and it minimized the cost of storage [8].

2.2 Reinforcement Learning Model, Algorithms and Its Application

Machine Learning is a branch of Artificial Intelligence which concerns learning of machines from data. One of the approaches of machine learning is through reinforcement learning. The study surveys the model and theory of reinforcement learning and presents the four main algorithms Sarsa, temporal difference, Q-learning and function approximation [9]. In a RL system there are 2 main components the agent and the environment and four essentials: policy, reward function, value function and model of the environment. All reinforcement learning algorithms involve estimating value of a state ($v\pi(s)$) or value of a state action pair ($q\ \pi\ (s, a)$). The functions give an estimate of how good it is for the agent to be in a given state or how good is the action that leads to the current state. The comparison of the algorithms is presented in table 1 as below. Reinforcement Learning is mainly used in process control, dispatch management, robot control, game competition and information retrieval etc., the widest application is the field of intelligent control and intelligent robot [9].

Table 1. Comparison of RL Algorithms

Sno	Algorithm	Complexity		Function
		Space	Time	
1	SARSA	Linear	Linear	Q-Function
2	Temporal Difference	Linear	Linear	Value Function

3	Q-Learning	Linear	Linear	Q-Function
4	Function Approximation	Depends	Depends	Construct from data

2.3 A RL approach to Virtual Machine Auto-configuration

VCONF [10] is a RL based approach for the automation of the VM configuration process. Virtual machines enable multiple machines to share the same hardware resources. Based on the demand of the application executing in the virtual machine the resources of the VM have to be dynamically reconfigured. The state in the RL system is the hardware configuration of the VMs. Actions are the change in the configurable parameters. The reward signal in the VCONF is based on the summarized performance of each VM [10]. The VCONF implemented a basic look-up table based Q function [10]. The system automated the VM reconfiguration process by generating policies learned from iterations with the environment. The VCONF performance was tested on both a controlled environment as well as cloud test bed. The workload was given in the form of E-commerce, OLTP and web server. In the presence of workload dynamics, VCONF was able to adapt to a good configuration within 7 steps and showed 20% to 100% throughput improvement over basic RL methods [10].

2.4 A RL framework for online data migration in hierarchical storage systems

In multi-tier storage systems, a very important performance indicator is the response time of the file I/O requests. The response time can be minimized if the frequently accessed files are located in the fastest storage tier. The challenge is that the file access patterns change over time. David Vengerov [11] proposed a framework that minimizes the average request response time. It is achieved by a RL agent that learns to tune the file migration policies. File migration is the process of displacing the file storage location to other tiers based on the file's access frequency. The migration rules are formed as policies which associate every situation to an action. The proposed framework uses a Reinforcement Learning (RL) methodology for tuning parameters of tier cost functions based on which file migration is performed. A temporal difference (TD) method is employed to learn the policy cost for all states of the system. The file migration is I/O triggered. The RL methodology applied here is the I/O triggered migration policy. The RL methodology was applied to tuning the parameters of the fuzzy rule base, which resulted in a 35% performance improvement relative to the best found pre-specified file migration policy [11].

Hierarchical Storage Management (HSM) effectively utilizes the storage space and meets customer demands. Data migration seamlessly moves data across tiers as the value of data changes over time. There are several shortcomings in the existing data migration techniques: (1) policies are user defined and hence static and reactive and (2) data migration at the host side is not yet completely explored. It has been observed in the past that RL framework has been used to perform data migration. But the approach has certain constraints: (1) each storage tier is modeled as an agent and the data migration is a joint decision between the 2 agents; (2) the data migration methodology is I/O triggered; and (3) tier cost represented as a complex fuzzy rule base (FRB). To address the above inadequacies this paper proposes to study the effect of a "Proactive and Adaptive Data Migration in Hierarchical Storage Systems Using Reinforcement Learning". As a part of implementation a single data migration agent in a Hierarchical Storage System (HSS) was developed and set up. The data migration agent is a standalone daemon which is proactive and adaptive. The proactive characteristic will ensure that the agent keeps in check the average response time of all storage tiers and perform data migrations before the waiting time deteriorates. Correspondingly the adaptive nature of the agent will adjust the data migration rate to avoid QoS violations for the applications. The above characteristics will be achieved by implementing the RL algorithm (Q-Learning) in the agent which will formulate and tune policies based on which the data migration will take place.

3. RESEARCH QUESTIONS

This paper aims to answer the following research questions:

(i) Does the RL agent learn to perform data migration when the average queue size of a device increases?

(ii) Does the RL agent reduce the queue size of devices by performing data migration to the appropriate devices?

(iii) Does the RL agent based data migration have no or zero effect on the write and read performance at the host end?

4. REINFORCEMENT LEARNING FOR DATA MIGRATION IN HSS

A single data migration agent in a Hierarchical Storage System (HSS) in Linux OS was implemented [12]. The data migration agent is a standalone daemon. The agent performs migration proactively thereby keeping in check the average response time of all storage tiers. The average response time is controlled by monitoring the average queue size. The data migrations will be performed before the waiting time in the storage tiers deteriorates. The agent would provide interfaces to adjust the data migration rate to avoid QoS violations for the applications. The agent utilizes the Q-Learning algorithm to formulate and tune policies based on which the data migration will take place. Figure 3 illustrates the bird's eye view of the environment and Figure 4 the organization of the RL agent.

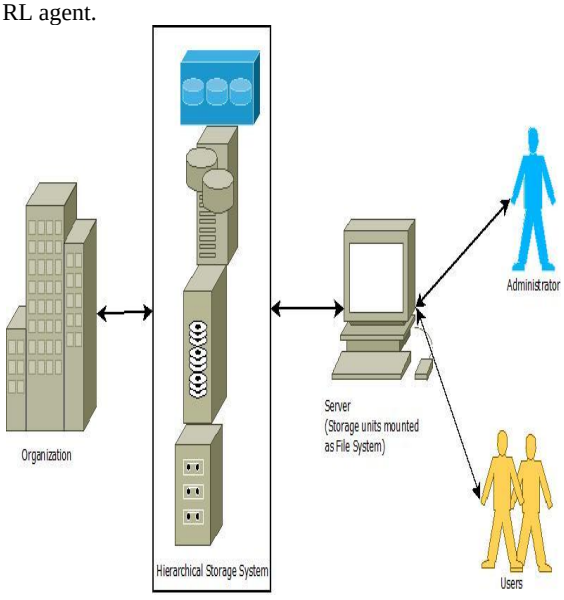

Fig 3: System Design

A storage tier is chosen as a candidate for data migration by the agent based on the values of state variable. The state variables of each storage tier are based on the notion of temperature of each file, which approximates it access rate [12]. The three state variables formed: (i) S1- no of transfers per second (tps), (ii) S2 - no of read/write requests per second (r/s &w/s) and (iii) S3- the average queue size of the device (avgqu_sz). Before the devices can be monitored, the hardware abstraction layer (HAL) in Linux is utilized to ascertain the no of devices and the file system mount point of the devices.

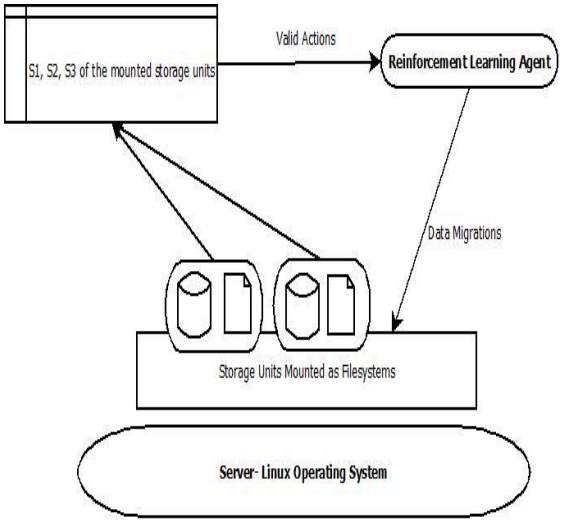

Fig 4: RL Agent Organization

4.1 Migration Environment

The migration environment keeps monitoring the 3 state variables tps, r/s&w/s and avgqu_sz using the command iostat. The iostat command is used to monitor the system input/output devices. It generates reports which can help us perform load balancing between the devices. The values of S1, S2 and S3 will help us identify the tier under load. The command iostat gives the value of the three state variables. The values are recorded with timestamp so as to track the improvement/decline of the state variables as a result of migration.

4.2 Migration Agent

The agent looks at the values of the state variables at regular intervals. If it is found that a particular tier is under a heavy load, that tier becomes a candidate for data migration. The RL algorithm updates the data migration rules and stores them as CRON entries. The cron is software utility in UNIX. It is a time-based job scheduler. Cron is a daemon that executes scheduled commands. Administrators use cron to schedule job to run periodically at fixed times, dates or interval.

Crontab is the program used to install/uninstall/run the cron daemon. When the minute, hour, and date match the current time commands are executed by cron. This helps us to initiate data migration at appropriate time so as to not disrupt the application performance. The "data migration rate" is determined by the administrator. It determines the frequencies at which the data migration can be triggered by the agent The CRON entries are triggered to perform the data migration. The table 2 depicts the algorithm for the RL agent.

Table 2. Data Migration RL agent

Algorithm RL agent for Data Migration
1: Initialize the devices using HAL
2: Initialize t ←0
3: repeat
4: s_t ← get_current_state( )
5: r_{t+1} ← observe_reward( )
6: a_{t+1} ← get_next_action( )
7: s_{t+1} ← perform action(a_{t+1})
8: t ← t+1
9: until RL agent terminates

The RL-based agent keeps track of the values of the state variables after a data migration has happened. The state variables with most recent timestamps can be compared to find out if the migration task has performed well. If a deterioration of the values is observed a penalty is added to the migration rule, similarly the migration rule which results in improvement of the observed values is rewarded. The migration rules are stored as simple lookup table. Based on the rewards/penalty the agent modifies the destination tiers in the CRON entries. After some trials, the agent would use the knowledge it learnt to perform near optimal migrations which improves the average I/O request response time of all the storage tiers. The figure 5 illustrates the implementation components along with the tools used to implement them.

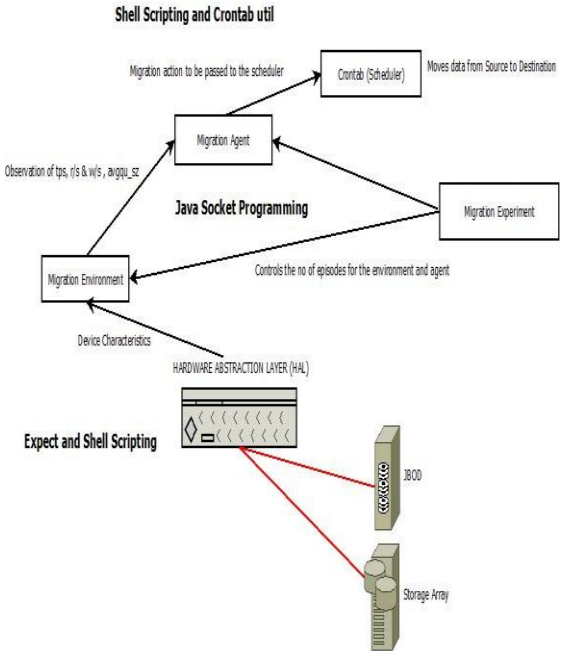

Fig 5: Implementation components

5. RESULTS

A single host (Ubuntu 12.04 LTS, 32-bit) with Intel dual CPU at 2.17 GHz, 2.00 GB RAM was connected to 3 devices each of 222 GB HDD, 8 GB USB and 4 GB USB devices. The transfer rates of each of the device are available in table 3.The RL agent was installed onto the host.

Table 3. Test Bed Setup

Sno	Storage Tier	Device	Transfer Speed (MBps)	Storage (GB)
1	Tier -I	ATA HDD	300	222
2	Tier -II	USB	35	8
3	Tier -III	USB	35	4

5.1 Test for average queue size of devices

A script was developed which in each of the device does the following in a loop:

- Create folders from A-Z
- In each folder creates files from A-Z of 5MB size each
- Deletes all folders and files

The above script was executed before RL agent is invoked and average queue size of all the 3 devices tabulated for about 1000 observations. The same script performed during RL agent execution and average queue size of all the 3 devices tabulated for about 1000 observations. The average queue size before and after the RL agent execution were compared for all the three devices. In total there were about 1000 observations, and in all the three devices with RL agent execution the maximum queue size was about 2.5, whereas the maximum queue size before the RL agent went to about 8, 22 and 32 in each of the three devices. The Figure 6 presents the comparison of the average queue size of a device without RL agent (in blue) and with RL agent (in red). This validates the premise that the RL agent improves the average queue size by performing proactive data migration. Figure 7 contains the impact on the average queue size of a device for 1000 observations. The graph shows that after about 500 observations, the average queue size of the device almost becomes negligible.

5.2 Test for RL agent effect on write/read performance

I/O load using iozone was created and output saved to excel worksheet. The measurement of file response characteristics (kbytes/sec) also monitored using tool iozone. The command used "iozone -a -b <name of excel file>.xls -i 0 -i 1 ". The above experiment was performed before RL agent is invoked and results tabulated. The same experiment is performed during the RL agent execution and results were tabulated.

A t-test was performed on the values of file response characteristics (Kilobytes per second) which were collected to measure the write/read performance for every file size on the paired sample group (before and during the RL agent execution). The t-test checks if the 2 means are reliably different from each other. In our case we do not want the means to be different from each other, as we would want to prove our hypothesis that the RL agent does not impact the I/O performance. The figure 8 tabulates the t-test results.

6. DISCUSSION AND CONCLUSION

The analysis of the results obtained in the previous section (section 6) leads to the answers of the research questions formulated in section 3. The answers are as follows:

- In 960 observations the average queue size of the 3 devices is about 4.301001
- With RL agent , in 960 observations the average queue size is reduced to 0.033069
- After an average of 500 cycles the RL agent brings down the average queue size of the device (avgqu-sz) to almost zero
- At 95 % confidence level, the RL agent does not affect the write performance of file sizes 128, 256, 512, 1024, 2048, 4096 and 8192 kilobytes
- At 95 % confidence level, the RL agent does not affect the read (with buffer) performance of file sizes 64, 128, 256, 512, 2048, 4096 and 16384 kilobytes

The data analysis that we have performed on the collected data in this paper answers all our research questions in affirmative. A proactive and adaptive data migration RL agent whilst performing data migration will be able to reduce the migration and data mobility costs. Though these costs may be soft in nature, but nevertheless it will be good enough to lower the total OPEX (Operational Expenses). A proactive and adaptive data migration technique at the host end of SAN would also improve the effective utilization of storage tiers and thereby enhance the TCO (Total cost of Ownership) of the storage components.

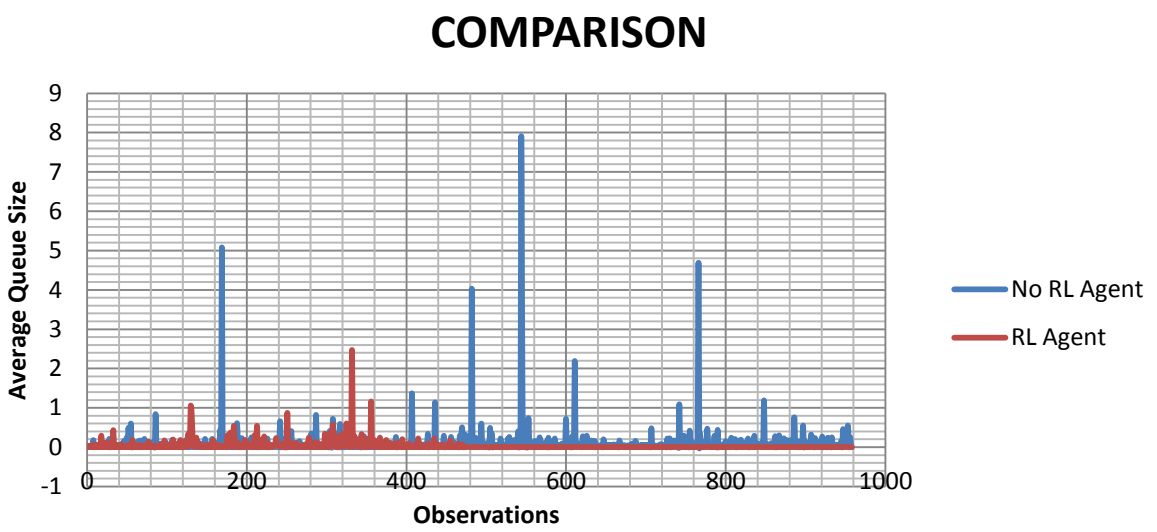

Fig 6: Comparision of average queue size in device 1

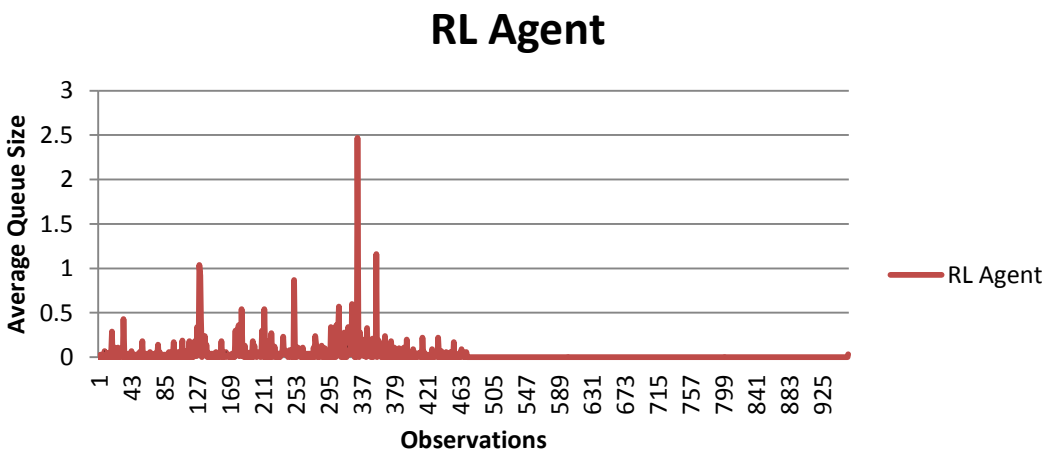

Fig 7: RL agent impact on the average queue size of device 1

To expand the study repetition of the experiment, at both content and period level would need to be performed. To further investigate the effect and study the impact better integration and implementation of the code on real life system HSS needs to be performed. This system can be further improved by:

- investigating the performance impact on read /write of file sizes greater than 16 MB
- reduction in the cycles taken by the RL agent to get a positive reward.

VCONF [10] was able to adapt to a good configuration within 7 steps and showed 20% to 100% throughput improvement over basic RL methods. In the model proposed by David Vengerov [11] it was found that statistically significant performance improvement was achieved only during the first three iterations of the learning cycle, reducing AWRT by 35% in comparison to the initial policy. The proposed system has comparable performance in terms of reducing the queue size of devices, but the number cycles taken to do so can be further deliberated. The study can be further broadened by the addition of the "Predictive" edge to the data migration strategy, where the agent can also predict based on learning what the data/device temperatures will be.

Writer			
Sno	File Size (In Kbytes)	No of Observations	Difference in Means (from t-test at 95 % confidence level)
1	64	5	Yes
2	128	6	No
3	256	7	No
4	512	8	No
5	1024	9	No
6	2048	10	No
7	4096	11	No
8	8192	12	No
9	16384	13	Yes

Reader			
Sno	File Size (In Kbytes)	No of Observations	Difference in Means (from t-test at 95 % confidence level)
1	64	5	No
2	128	6	No
3	256	7	No
4	512	8	No
5	1024	9	Yes
6	2048	10	No
7	4096	11	No
8	8192	12	Yes
9	16384	13	No

Fig 8: t-test results of I/O performance

7. ACKNOWLEDGMENTS

Our thanks to the principal of Thakur College of Engineering and Technology, Dr.B.K.Mishra for his constant motivation and support.

8. REFERENCES

[1] Somasundaram, G., & Shrivastava, A. (2009), "Information storage and management", Wiley Publishing, Inc.

[2] Clark, Tom (2003), "Designing Storage Area Networks: A Practical Reference for Implementing Storage Area Networks", Addison-Wesley Longman Publishing Co., Inc.,

[3] Morse, E. A., & Raval, V. (2008), "PCI DSS: Payment card industry data security standards in context", Computer Law & Security Review, 24(6), 540-554.

[4] IBM Redbook(1999), "Hierarchical Storage Management - AS/400e" IBM

[5] Sutton, R. S., & Barto, A. G. (1998), "Reinforcement learning: An intro-duction", Cambridge, MA: MIT Press

[6] L. P. Kaelbling, L. M. Littman, and A. W. Moore (1996), "Reinforcementlearning: A survey," Journal of Artificial Intelligence Research, vol. 4, pp. 237–285

[7] Puterman, M. L. (2009),"Markov decision processes: discrete stochastic dynamic programming" (Vol. 414), New Jersey, John Wiley & Sons.

[8] Lodzimierz Funika, W., & Szura, F. (2013), "Data Storage Using AI Methods". Computer Science, 14, 2.

[9] Qiang, W., & Zhongli, Z. (2011, August), "Reinforcement learning model, algorithms and its application", In Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference on (pp. 1143-1146). IEEE.

[10] Rao, J., Bu, X., Xu, C. Z., Wang, L., & Yin, G. (2009, June), "VCONF: a reinforcement learning approach to virtual machines auto-configuration", In Proceedings of the 6th international conference on Autonomic computing (pp. 137-146). ACM.

[11] Vengerov, D. (2008)," A reinforcement learning framework for online data migration in hierarchical storage systems" The Journal of Supercomputing, 43(1), 1-19.

[12] Lakshmi T.G., Sedamkar R.R., Patil H (Nov.-Dec., 2013), "Reinforcement Learning Approach for Data Migration in Hierarchical Storage Systems", International Journal of Enhanced Research in Management & Computer Applications, ISSN: 2319-7471,Vol. 2 Issue 9, pp: (30-35)