Enabling proactive data management in virtualized Hadoop clusters based on predicted data activity patterns

George Kousiouris, George Vafiadis and Theodora Varvarigou
Department of Electrical and Computer Engineering
National Technical University of Athens, Greece
gkousiou@mail.ntua.gr, gvaf@iccs.gr, dora@telecom.ntua.gr

***Abstract*— Hadoop clusters are gaining more and more in popularity based on their ability to parallelize and complete large scale computational tasks on big data. Service offerings of this type have appeared in the recent years, covering a need for dynamic and on-demand creation of such data analytics frameworks. The aim of this paper is to provide a mechanism for offering such virtual clusters as a service, with built-in intelligence functionalities for efficient management. The target of these mechanisms is to predict future demand of the files in the HDFS cluster and dynamically manipulate the according replication factor for availability purposes, in order to improve performance and minimize storage overhead. To this end, real data have been utilized as a dataset input to the prediction framework, based on Fourier series analysis, due to the latter's ability to capture different periodicities that can influence service usage. Multiple time-step ahead prediction is performed in order to enable proactive management (e.g. suitable replication strategy). We describe the framework's architecture, necessary modifications to the client side of Apache Hadoop for data logging and the results of the applied method on two real world datasets.**

Keywords—Big data; proactive replication; time series prediction; dynamic management; Fourier series analysis

I. INTRODUCTION

As produced and uploaded data over the Internet grow, it becomes more and more difficult for applications and infrastructures to manage, process and extract meaningful and added value information from them. To this direction, massively parallel and distributed computing paradigms have emerged, such as the MapReduce framework[1]. One key feature of these solutions is the replication of the data items, either for reliability and fault tolerance purposes or for improving availability and performance.

However, the application of replication techniques implies that a large part of the disk space (the exact percentage depends on the replication factor) will be utilized for maintaining the same data, thus resulting in poor actual utilization. Furthermore, while the performance benefit in read operations would be improved by the existence of multiple copies, in cases of write operations the necessary synchronization between them would result in a performance deterioration. Thus a static replication factor for the entire distributed file system and all the included files may result in poor management, under-performance and extensive over-provisioning.

In this paper a framework that enables automatic and proactive management of an Apache Hadoop and HDFS[3] instance offered as a service is portrayed. The framework is able to keep the necessary detailed statistics of the data access usage through a modified version of the main client for accessing HDFS instances,sshFS[2], effectively compress them through a MapReduce task and feed them in a service oriented time series prediction component. The latter creates prediction models based on the previous access values and is able to identify the data items that are expected to be more popular in the future (for an extended look-ahead interval). Thus it may be used for driving the configuration of a suitable replication factor that will balance between increased availability and disk usage. By over-replicating the future anticipated popular data (also known as "hot") and under-replicating the popularity-declining ones (also known as "cold"), we can achieve a more balanced and optimized file system configuration. This process is iterative since depending on the data access patterns, an item may move from a hot to a cold status over time and vice versa.

The remainder of the paper is structured as follows. In Section II related work relevant to the addressed topic is presented. In Section III an architectural overview of the system and its design is portrayed while in Section IV the specifics of the time series prediction mechanism are included. In Section V the accuracy of the prediction framework is investigated through two case studies, while Section VI concludes the paper.

II. RELATED WORK

A number of interesting relevant works can be found with regard to the concepts expressed in this paper. Numerous distributed storage systems with different characteristics have evolved during the past years [17]. CDRM[5] is a framework to automatically calculate and place replicas in a cloud data store based on workload changes, in order to optimize data availability while reducing used space. However the popularity of a file is measured in a reactive process. After the file

accesses reach certain thresholds, the popularity level rises and then suitable increase in replicas is performed. In our work, we predict proactively this change, and then the CDRM placement strategy may be applied complementary for replica number and placement decision on the nodes. This a priori knowledge is useful in cases where the preparation time for the cluster is significant, e.g. due to the involved data size. Due to the latter, replication time (the time to actually create and distribute the replicas) may be significant. Thus with a reactive approach, once the popularity threshold is passed and the replication kicks off, by the time it finishes the popularity levels may have changed again.

Analysis of differences in file access patterns is also performed in [7], indicating the fact that file popularity may vary and thus uniform replication strategies are not sufficient. In this case, prediction of file popularity is made to some extent, however based on the examined dataset properties, not involving automated and on-demand model creation for different access patterns. The benefit of applying dynamic replication is evident also in this approach. Improving data locality with various ways (including investigation of replication effects) is also presented in [9]. Usage patterns are furthermore discussed in [16] where a usage model is used to decide on the placement of storage objects.

In [10], an interesting approach is presented for correlating anticipated file access (and thus apply suitable management plans for performance and energy efficiency) which takes under consideration observations from a real world dataset. The latter indicate that the directory structure is a very crucial factor in the anticipation for file accesses. Correlation-based prefetching is also applied in [11].

The issue of replicating across different HDFS clusters and saving disk space from redundant replications is dealt with in HotROD[7]. Especially interesting is the fact that cold data are degraded to zero replication and shared between clusters in order to minimize disk space. In case they need to be accessed by one of the clusters, this is done over the shared location. This is especially interesting when combined with virtualized Hadoop clusters that may be federated across different cloud providers, as is performed in [13] and [8]. This federation may be performed for a variety of reasons such as meeting internal resource scarcity or risk management.

From the examined related work we can conclude that while numerous very interesting and effective works exist for determining and managing optimal replica numbers and placement, little attention has been given to the dynamic proactive determination of what is considered cold and hot. File popularity in most cases is determined reactively or based on general conclusions on examined datasets properties. Furthermore, with the rapid uptake of cloud services (and specifically MR virtual clusters), the use cases for cloud services may extend to more complicated usage patterns than in standard Apache Hadoop environments. Thus the dynamic popularity prediction mechanism described in this paper can improve the capabilities of the examined systems, especially in cases where the replication enforcement delay (actual time to create the replicas across the nodes) is not trivial.

III. FRAMEWORK ARCHITECTURE AND SYSTEM DESIGN

The OPTIMIS Distributed File System (ODFS) solution is heavily based on the Apache Hadoop framework, and mainly its underlying file system, HDFS. The purpose of the ODFS is to offer virtualized Hadoop clusters as a service, to accompany regular service VMs and act as a storage and data analytics cluster. In order to offer it as a service, the two main node types, the Master Node (Hadoop Namenode) and the Slave Node (Hadoop Datanode) have been incorporated in virtual machine (VM) templates. A suitable RESTful gateway (the OPTIMIS DM) has been implemented in order to enable the creation and management of the data clusters during startup and runtime. The DM is responsible for acquiring requests from the OPTIMIS Infrastructure Provider (IP) in order to launch a group of these data VMs for a specific service. Following these requests, the DM launches for this service ID a Namenode, accompanied by the suitable number of Datanodes, that consist of the virtualized distributed file system and data analytics framework. Afterwards, the regular VM servers that host the applications (service VMs) are contextualized and a suitable client is installed in order for the users to mount and use the ODFS data VMs.

As stated in the introduction, what would be of interest is the ability to manage and adapt during runtime such virtualized data analytics offerings, in order to optimize their behavior, reduce operational costs and improve performance. Towards this end, in the context of the OPTIMIS project we have developed a suitable subsystem for monitoring, logging and runtime prediction acquiring framework for usage in Hadoop (virtualized or not) clusters. This system has the purpose of documenting user accesses towards the ODFS, keeping statistics, creating and using time series prediction models in order to anticipate peaks in data activity. The generic component diagram for the overall DM solution appears in Figure 1, including aspects such as federation, legal aspects of data management and risk management. In this paper we will focus on the FCS/DAF component and the logging subsystem.

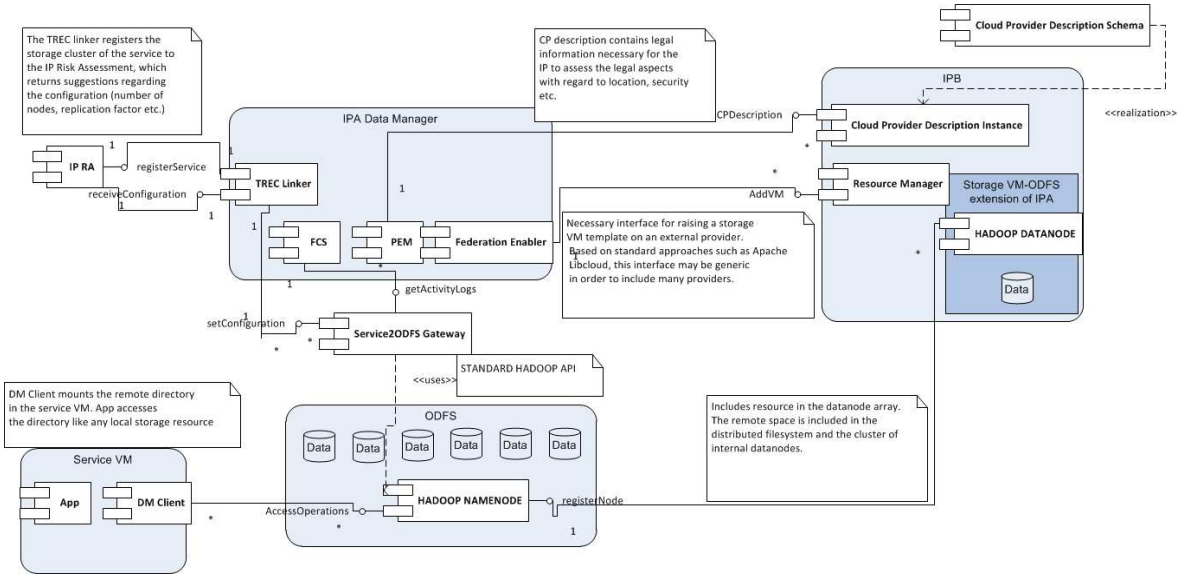

Figure 1: Generic Component Diagram for OPTIMIS DM

The sequence diagram for the logging subsystem and operation appears in Figure 2. In this case the LoggingMonitor subscribes to the DM. Then the ServiceODFSGateway (helper component) sends the logs to the Monitor which stores them in the HDFS as a distributed file. From the DM GUI web portal an interested party can then trigger a MapReduce (MR) job in order to retrieve the compressed statistics of data I/O operations on this instance of the data service. The compression is necessary for two reasons:

- Reduce the required disk space for the logs since
- Preprocess them to the required form for the prediction algorithm model (as it will be seen in the following sections, the prediction method needs average I/O values in time intervals which smoothens the graph and makes the prediction feasible)

In case it is needed to maintain a response time independent of the MR completion time, this job may be scheduled periodically (e.g. after the end of each time interval of the time series) so that the statistics are available upon request. An example of compression ratio for the logs has been applied on the MSR Cambridge traces, where the reported values are detailed in the form <timestamp>, <operation>, <size of operation> etc. For the compression we used a 15-minute interval to extract average values of read/write size operations which compressed the logs from an initial 8.3GB of data to 88.3 KB. Furthermore, due to the fact that prediction methods usually require a specific range of previous time steps (e.g. 10 15-minute average previous steps), the older log values may be deleted in order to further minimize disk space. Thus the storage overhead of the logging mechanism can be considered as negligible.

In Figure 3 the sequence for the prediction acquisition appears. An interested party (e.g. cloud management framework) can acquire predictions for the data activity of a specific service ID through the use of the DM client's relevant method (getDataActivity). The user must specify the service ID for which the activity should be forecasted along with the lookAhead interval. This is the number of future steps that need to be predicted. How long this is in wall clock time depends on the used time intervals in the prediction model (usually each step in the time series is the average I/O in the next 15 minutes). Afterwards the DM client contacts the Data Activity Forecaster component in order to acquire necessary information regarding the prediction model, in terms of needed past historical values. Then the DM client contacts the logging subsystem in order to get these values and call the getPrediction method of the DAF with all the necessary information (past values, model ID and needed future predictions).

A. DAF SERVER architecture

The DAF server architecture is a layered service, with a RESTful interface. It is encapsulated in an application server (like Tomcat or Glassfish) and utilizes Matlab-based executable plugins for model creation and querying, following the design of [14]. Thus the underlying models and methods may be easily replaced. The server was migrated on the Amazon AWS EC2 offering. The purpose for the latter was to investigate the performance improvement after the addition of the Amazon Elastic Load Balancer in front of the mechanism. This would make it feasible for the service to be scalable in case of increased traffic. Furthermore, an experiment was conducted in order to correlate the number of VMs deployed with the desired QoS (server response time). More information on these experiments is included in Section V.

B. Envisaged Use Cases

The benefit of the aforementioned mechanism can be applied in the following cases.

1) Selection of service VMs to federate

In a multi-provider environment [13], the ability to federate from one IP to another is crucial in terms of achieving

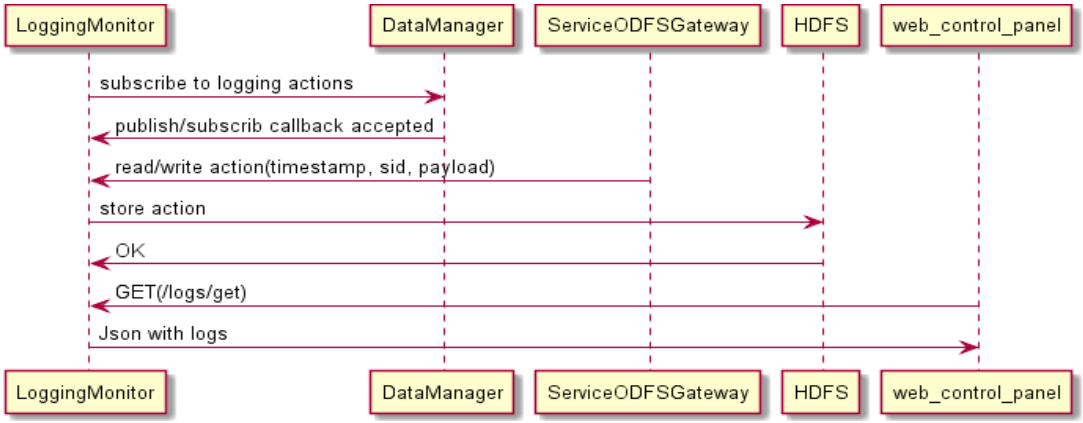

Figure 2: Sequence diagram of logging information

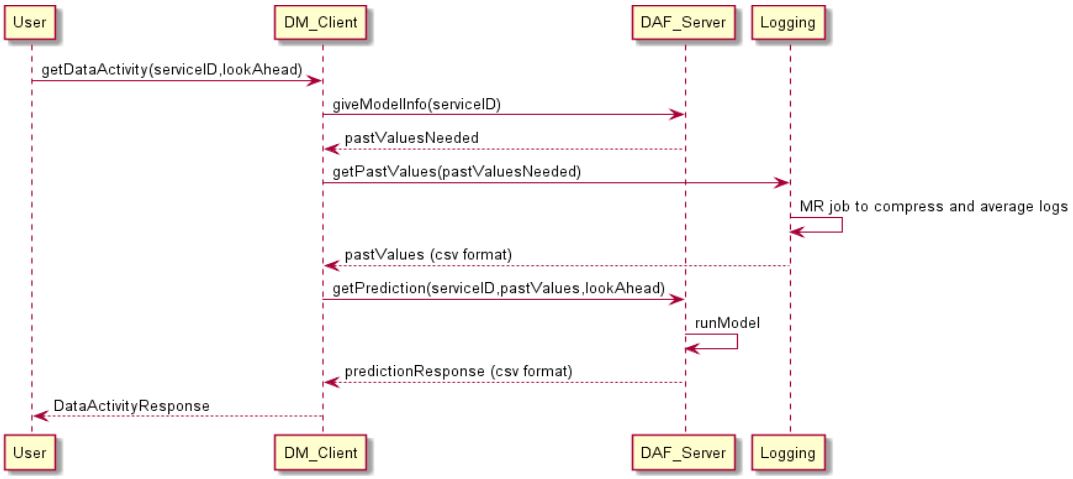

Figure 3: Sequence diagram for acquiring data activity forecasts

availability obligations or enabling probabilistic overbooking strategies. Furthermore, in bursting strategies (from a private cloud to a public one) this can also apply for meeting peaks in demand.

However in these actions, data would be kept in-house either for confidentiality issues (bursting of private cloud) or for performance ones (increased delay in moving large amounts of data from one provider to the other for a federation that may potentially last for a small amount of time to meet specific loads). Thus by using the presented mechanism a provider may rank the candidate service VMs for federation based on their anticipated activity towards the virtual Hadoop data cluster that will remain in-house. Through this, the ones that will exhibit the least activity can be selected, enabling the minimization of the performance overhead from accessing data over a remote network location (from the federated or bursted provider).

2) Dynamic Replication Management for increased availability and minimized disk usage

Knowing in advance the demand for data may also lead to enabling optimal replication management schemes. The latter may aid in two areas. Initially, increased number of replicas for anticipated "hot" data will result in more sources being able to accommodate incoming requests, thus improving performance and response time (especially in read operations). What is more, reduced number of replicas for predicted "cold" data may result in reduced disk usage. Following these predictions, suitable management schemes (like CDRM or ERMS that were investigated in Section II) can be applied in order to rearrange aspects such as the number and location of replicas.

IV. Time Series Prediction Component for Data Activity Patterns

The available input is the compressed time series of average I/O values in a predefined time interval, in terms of MB/second. This is done as seen in the previous section, for compressing the logs but also for hiding large variations in the

time series itself. The output (prediction) values can be fed back to the input in order to iterate the prediction and thus extend it to multiple time steps ahead. A typical time step ranges between 15 minutes and 1 hour in the cases of the time series we have investigated.

Fourier Series Analysis

Due to the fact that the examined data displayed a periodic nature, a classic method from the signal analysis domain (Fourier series analysis) was selected. Analysis using the Fourier factors is used in order to depict the effect of various frequencies of a signal in the overall signal function. This way the main signal can be separated from the noise. In our case it was used since it was considered a way of identifying different workload patterns (with different frequencies) that may affect the overall data demand (for example, the main period of the data demand "signal" can be considered the working week, while larger periods of influence may indicate specifics such as summer/christmas holidays etc.). The function that was used in order to depict the periodic demand signal appears in .

f(x)= a0 + a1*cos(x*w) + b1*sin(x*w) + a2*cos(2*x*w) + b2*sin(2*x*w) + a3*cos(3*x*w) + b3*sin(3*x*w) + a4*cos(4*x*w) + b4*sin(4*x*w) + a5*cos(5*x*w) + b5*sin(5*x*w) + a6*cos(6*x*w) + b6*sin(6*x*w) + a7*cos(7*x*w) + b7*sin(7*x*w) + a8*cos(8*x*w) + b8*sin(8*x*w)

Equation 1: Fourier series analysis function

The Matlab implementation of the function was used (fit function with "fourier8").

V. Evaluation Case studies

In order to evaluate the proposed method, we used two traces from real data in order to acquire time series from actual services demand. In the first case, we utilized a trace from a three month period of Wikipedia demand per hour. The reason for choosing this type of trace is to investigate the fluctuations in the workload, which in most cases result in fluctuation in the read/write operations, of a popular service offering. For the second experiment we utilized sample data provided from the Cloud Provider Arsys[4], with regard to their platform (sample from the email offerings). For this case, we acquired the actual I/O of the service towards the file system, in order to observe directly the fluctuation in read/write operations over time.

For both experiments, the available datasets have been divided in two parts, 70% used for training and 30% for validation. It is necessary to stress that during validation we perform multiple step-ahead prediction, meaning that the models start predicting at the first validation value and continue predicting without utilizing the actual validation values. These are only used in the final comparison of the predicted values against the actual ones. As a performance metric we use the Mean Absolute Error (Equation 2), since it is a stricter metric than the mean error.

$$MAE = \frac{\sum_{i=1}^{K}[\frac{|f(x_i) - Y_i|}{|Y_i|}]}{K} * 100$$

Equation 2: MAE definition: K number of samples, f estimated output, Y real output

A. Wikipedia experiment

The Wikipedia trace extends from July to September 2011 and consists of 2208 values (site hits per hour). The training set aids in determining Equation 1 parameters, that appear in Table 2.

The validation sample is 30% of the dataset, thus 0.3*2208=662 values. The results appear in Table 1. The fact that the errors do not keep increasing with the time steps is that the curves have been adapted to the main periodicity, so potentially for an interval that does not follow this pattern the errors may appear increased (in the Wikipedia dataset for example this seems to be between the 100th and 150th validation points).

The results for the Fourier Series Analysis appear in Table 1, while the fitting parameters in Table 2.

B. Arsys Mail server data pattern

The Arsys trace includes a period from September 2012 up to April 2013. It captures the average data accesses (in bytes) of a mail service towards the network file system in 15-minute intervals. The overall trace consisted of 23232 values.

We applied this trace to the Fourier analysis method, by using 70% of the dataset for training and the remaining 30% for validation. It is necessary to stress that during validation the prediction begun at the first validation sample and no feedback was provided to the model. Thus the prediction carried on for nearly 7000 look ahead values in the future. Results appear in Table 3, for a variety of groupings of the look-ahead errors, and overall in Figure 4. The coefficients of the Fourier formula are included in Table 4.

Table 1: Error Table of Estimation Methods in the Wikipedia Dataset for a variety of step ahead predictions

Estimation Method and Steps Ahead	Mean Absolute Error (%)	Maximum Absolute Error (%)	St. Deviation of absolute percentile errors
Fourier8 10 steps ahead	14.29	24.49	5.93
Fourier8 50 steps ahead	9.44	24.7	6.33
Fourier8 100 steps ahead	12.37	32.72	9.03
Fourier8 150 steps ahead	12.82	32.72	8.78
Fourier8 overall (662 steps ahead)	11.03	35.33	8.01

Table 2: Fourier fitting parameters for Wikipedia dataset

Fourier Function Parameters	Value	95% Confidence Bounds
a0	87.3	(86.97,87.63)
a1	-1.874	(-2.344,-1.404)
b1	-4.515	(-4.987,-4.043)
a2	2.155	(1.684,2.626)
b2	0.01323	(-0.4572,0.4837)
a3	0.2445	(-0.2258,0.7148)
b3	0.7233	(0.2528,1.194)
a4	0.2024	(-0.2677,0.6725)
b4	-0.3269	(-0.7971,0.1433)
a5	-0.8844	(-1.355,-0.4137)
b5	0.5182	(0.04753,0.9889)
a6	-0.4376	(-0.9092,0.034)
b6	-0.8254	(-1.296,-0.355)
a7	6.617	(5.747,7.487)
b7	-13.85	(-14.43,-13.27)
a8	-1.082	(-1.552,-0.6111)
b8	-0.01591	(-0.4921,0.4603)
W	0.03731	(0.0373,0.03732)

Table 3: Error Table of Estimation Methods in the Arsys Dataset for a variety of step ahead predictions (15 minute, 6969 predicted values)

Estimation Method and Steps Ahead	Mean Absolute Error (%)	Maximum Absolute Error (%)	St. Deviation of absolute percentile errors
Fourier8 10 steps ahead	14.34	31.65	8.22
Fourier8 50 steps ahead	23.02	74.85	18.13
Fourier8 100 steps ahead	53.08	249.91	57.99
Fourier8 150 steps ahead	43.72	249.91	49.74
Fourier8 overall	40.08	471.59	39.87

Transforming the time series to hourly values did not help in improving the results, since the errors were higher than the 15-minute values.

1) Retraining of the model

Another advantage of the Fourier analysis is the low computation that is needed for training and finding good values for the function parameters (indicatively this process takes around 20 seconds even for cases of large 20,000-value datasets). This makes it feasible to retrain the model at different periods in order to refresh the prediction ability and capture new seasonal patterns.

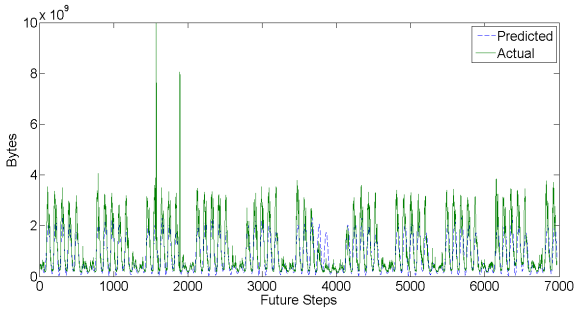

Figure 4: Overall graph of validation in the Arsys Dataset for a variety of step ahead predictions (15 minute, 6969 steps ahead)

Table 4: Fourier fitting parameters for the Arsys dataset (15-minute values, 69696 predicted values)

Fourier Function Parameters	Value	95% Confidence Bounds
a0	8.67E+08	(8.606e+008, 8.733e+008)
a1	-2.82E+08	(-2.908e+008, -2.727e+008)
b1	-3.35E+08	(-3.436e+008, -3.255e+008)
a2	7.43E+07	(6.516e+007, 8.337e+007)
b2	-2.19E+08	(-2.275e+008, -2.095e+008)
a3	8.56E+07	(7.655e+007, 9.457e+007)
b3	6.45E+05	(-8.401e+006, 9.691e+006)
a4	-9.52E+07	(-1.042e+008, -8.619e+007)
b4	-3.39E+07	(-4.301e+007, -2.487e+007)
a5	7.63E+06	(-1.679e+006, 1.695e+007)
b5	-1.73E+08	(-1.816e+008, -1.636e+008)
a6	2.46E+08	(2.364e+008, 2.546e+008)
b6	-7.55E+07	(-8.533e+007, -6.566e+007)
a7	-6.18E+08	(-6.299e+008, -6.061e+008)
b7	-4.13E+08	(-4.276e+008, -3.981e+008)
a8	3.07E+07	(2.033e+007, 4.11e+007)
b8	2.37E+08	(2.283e+008, 2.463e+008)
w	0.009347	(0.009346, 0.009347)

In order to demonstrate this process, we broke down the overall sample of 23,232 values to smaller periods, in which 50% of the dataset was used for training and 50% for validation. The period size was set to 4000 values, which is roughly 6 weeks of data. The results appear in Table 5. In some cases the errors seem quite high, this is due to some observed anomalies in the dataset (cases of extreme peaks that are not met before or after). Furthermore, another aspect is the fact that in most cases the prediction identifies with increased accuracy the peak pattern, however it loses the amplitude of the peak, as it can be seen in Figure 5. Indicatively, the overall prediction values for this case (last period of the retraining process, prediction for 1600 steps ahead) are as high as 38% average error, 258% maximum and 41% standard deviation. Despite the fact that these numbers seem quite high, it is evident from the graph that this is due to the peak miss and not the actual pattern. The validation sample consists of only 1600 values (and not 2000) since it is the last period of the 23,232 value data set, divided by 4000 value-period.

Table 5: Arsys periodic retraining every 4000 values

Estimation Method and Steps Ahead	Mean Absolute Error (%)	Maximum Absolute Error (%)	St. Deviation of absolute percentile errors
Fourier8 10 steps ahead (Period 1)	51.18	68.12	12.22
Fourier8 10 steps ahead (Period 2)	6.60	12.73	4.29
Fourier8 10 steps ahead (Period 3)	176.87	268.65	54.77
Fourier8 10 steps ahead (Period 4)	11.63	29.61	8.78
Fourier8 10 steps ahead (Period 5)	16.51	28.08	7.42
Fourier8 10 steps ahead (Period 6)	15.96	26.06	6.72

Also by using the actual values of the validation periods we can train the models without intervals, except for the initial one for getting the required data.

C. DAF server performance

In order to evaluate the DAF server performance and scalability boundaries, it has been installed in a medium instance Amazon VM (2 cores, 1.7 GB of RAM), running Ubuntu OS. In order to stress the server, a client simulator has been developed that has the purpose of launching requests. The threads are created simultaneously in order to simulate a burst mode of traffic, which is the worst case scenario. The number of calls is regulated accordingly, so that for each user number (5,10,20,30,40 concurrent users) 1000 calls are conducted. The results are gathered and processed so that the average call delay and standard deviation is collected. Initial and final 10% of the dataset is discarded, in order to avoid transient effects on the client side thread generation and termination and ensure that all the observed measurements are conducted for the desired number of concurrent users. For concurrent clients higher than 40 it was observed that the majority of the calls resulted in timeouts. The results for the average delay appear in Figure 6 and for the standard deviation in Figure 7.

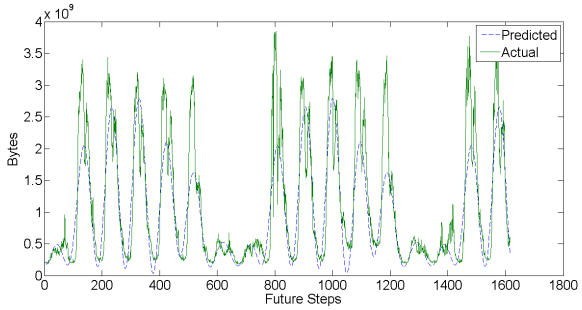

Figure 5: Last period of validation, indicative of loss of peak prediction

Furthermore, by comparing the results with [12], it can be observed that the Matlab-based implementation is significantly slower that an Octave based one. While the base times for one user are around 1.2 seconds for Octave, in the Matlab case this reaches about 6 seconds. The inner operation times (calculation of the forecast) in both cases are a very small percentage of the overall time delay.

Another aspect is the achieved scalability through the Amazon Elastic Load Balancer addition, that enables the deployment of multiple VMs. It can be observed that the 1 VM instance deployment scheme is not sufficient for higher number of users. Depending on the needed QoS offered by the server (in terms of the response time), we can regulate the number of VMs deployed in order to manage the incoming traffic.

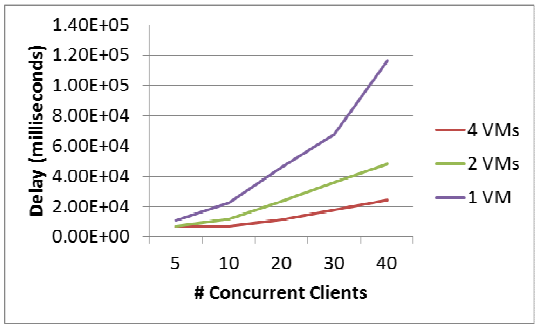

Figure 6: Average server response delay for varying number of concurrently requesting threads and number of deployed VMs

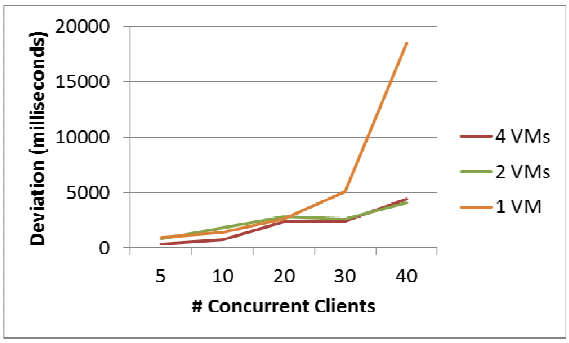

Figure 7: Average server response delay deviation for varying number of concurrently requesting threads and number of deployed VMs

VI. CONCLUSIONS

As a conclusion, Hadoop clusters are gaining more and more popularity as virtual dynamic offerings. Towards this direction, numerous interesting works exist that aim to optimally manage aspects such as replication factor, data availability etc. However these frameworks typically use current data popularity, as it is expressed through kept statistics. In this paper, a proactive management framework is presented, that can be used in combination with the aforementioned works in order to predict the data popularity in the future periods. This is an important optimization aspect, since when dealing with big data, actions to properly configure the infrastructure may be time consuming. Thus they should be performed in advance of the anticipated demand.

The proposed framework adds more logging capabilities to the standard Hadoop framework, without influencing the underlying APIs, and implements algorithms for capturing the access patterns of different real world datasets (namely a Wikipedia 3 month usage period and a web mail platform 8 month usage period from the Cloud provider Arsys). The ability refers to multiple step ahead predictions, without knowledge of the actual values in the validation period. Furthermore, the implementation boundaries of the prediction server have been investigated and additional architectural additions (such as the AWS ELB) have aided in extending the operational limits.

For the future, one aspect that needs to be investigated is the access patterns of additional types of services that may include different periodicities. Furthermore, it would be interesting to increase the number of Fourier factors in order to observe the effect on the solution's accuracy, thus investigating the trade off between generic patterns and overfitting. What is more, the usage of traces that extend even further in time (capturing years of usage) can help identifying more periodic phenomena (such as summer vacations etc.).

ACKNOWLEDGMENT

This research is partially funded by the European Commission as part of the European IST 7th Framework Program through the project OPTIMIS under contract number 257115 and through the project ARTIST under contract number 317859. The authors would also like to thank Arsys, for the provision of the platform sample.

REFERENCES

[1] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113.

[2] SSHFS client, Available at: http://fuse.sourceforge.net/sshfs.html

[3] Apache Hadoop, Available at: http://hadoop.apache.org/.

[4] Arsys Provider, http://www.arsys.net

[5] Qingsong Wei; Veeravalli, B.; Bozhao Gong; Lingfang Zeng; Dan Feng, "CDRM: A Cost-Effective Dynamic Replication Management Scheme for Cloud Storage Cluster," Cluster Computing (CLUSTER), 2010 IEEE International Conference on , vol., no., pp.188,196, 20-24 Sept. 2010

[6] Zhendong Cheng; Zhongzhi Luan; You Meng; Yijing Xu; Depei Qian; Roy, A.; Ning Zhang; Gang Guan, "ERMS: An Elastic Replication Management System for HDFS," Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on , vol., no., pp.32,40, 24-28 Sept. 2012

[7] Sriram Rao, Benjamin Reed, and Adam Silberstein, HotROD: Managing Grid Storage With On-Demand Replication, Workshop on Data Management in the Cloud (DMC'13), April 2013

[8] Tom Kirkham, Karim Djemame, Mariam Kiran, Ming Jiang, Django Armstrong, George Kousiouris, George Vafiadis, Athanasia Evangelinou "Risk Based SLA Management in Clouds A legal perspective", IEEE Internet Technology And Secured Transactions, 2012 International Conferece For , vol., no., pp.156,160, 10-12 Dec. 2012

[9] Zhenhua Guo, Geoffrey Fox, and Mo Zhou. 2012. Investigation of Data Locality in MapReduce. In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012) (CCGRID '12). IEEE Computer Society, Washington, DC, USA, 419-426.

[10] Kaushik, R.T.; Abdelzaher, T.; Egashira, R.; Nahrstedt, K., "Predictive data and energy management in GreenHDFS," Green Computing Conference and Workshops (IGCC), 2011 International , vol., no., pp.1,9, 25-28 July 2011

[11] Bo Dong; Xiao Zhong; Qinghua Zheng; Lirong Jian; Jian Liu; Jie Qiu; Ying Li, "Correlation Based File Prefetching Approach for Hadoop," Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on , vol., no., pp.41,48, Nov. 30 2010-Dec. 3 2010

[12] G. Kousiouris, A. Menychtas, D. Kyriazis, K. Konstanteli, S Gogouvitis, G. Katsaros, T. Varvarigou, "Parametric Design and Performance Analysis of a Decoupled Service-Oriented Prediction Framework based on Embedded Numerical Software", IEEE Transactions on Services Computing, 09 Aug. 2012. IEEE computer Society Digital Library. IEEE Computer Society

[13] A.J. Ferrer et al., "OPTIMIS: a Holistic Approach to Cloud Service Provisioning, , Future Generation Computer Systems, Elsevier, Vol. 28, No. 1, pp. 66-77, 2012.

[14] G. Kousiouris, D. Kyriazis, K. Konstanteli, S. Gogouvitis, G. Katsaros and T. Varvarigou, "A Service-Oriented Framework for GNU Octave-Based Performance Prediction," Services Computing (SCC), 2010 IEEE International Conference on , vol., no., pp.114-121, 5-10 July 2010,doi: 10.1109/SCC.2010.37

[15] G. Kousiouris, A. Menychtas, D. Kyriazis, S. Gogouvitis, T. Varvarigou, " Dynamic, behavioral-based estimation of resource provisioning based on high-level application terms in Cloud platforms" , Future Generation Computer Systems, Available online 29 May 2012, ISSN 0167-739X, 10.1016/j.future.2012.05.009.

[16] Spyridon V. Gogouvitis, Gregory Katsaros, Dimosthenis Kyriazis, Athanasios Voulodimos, Roman Talyansky and Theodora Varvarigou, "Retrieving, Storing, Correlating and Distributing Information for Cloud Management," 9th International Conference on Economics of Grids, Clouds, Systems, and Services, November 2012.

[17] Spyridon V. Gogouvitis, Athanasios Voulodimos and Dimosthenis Kyriazis, "Commercial and Distributed Storage Systems," Data Intensive Storage Services for Cloud Environments, IGI Global, 2013. doi:10.4018/978-1-4666-3934-8.ch001