Premonition of Storage Response Class Using Skyline Ranked Ensemble Method

Kumar Dheenadayalan*, Muralidhara VN†, Pushpa Datla*, G Srinivasaraghavan† and Maulik Shah*
*Qualcomm India Pvt Ltd. Bangalore, India
†International Institute of Information Technology-Bangalore, India

***Abstract*—Tertiary storage areas are integral part of compute environment and primarily used to store vast amount of data that is generated from any scientific/industry workload. Modelling the possible pattern of usage of storage area helps the administrators to take preventive actions and guide users on how to use the storage areas, which are tending towards slower or unresponsive state. Treating the storage performance parameters as a time series data helps to predict the possible values for the next *'n'* intervals using forecasting models like ARIMA. These predicted performance parameters are used to classify if the entire storage area or a logical component is tending towards unresponsiveness. Classification is performed using the proposed Skyline ranked Ensemble model with two possible classes, *i.e.*, high response state and low response state. Heavy load scenarios were simulated and close to 95% of the behavior were explained using the proposed model.**

***Keywords*-storage response time; ARIMA; Ensemble method; Skyline query; Random Forest; SVM; SOM;**

I. Introduction

Storage is a key component in any Distributed/Grid/Cloud Computing environment. Most scientific simulations are I/O intensive applications, which are complex in nature. Such tasks are solved on a large scale using Grid/Cluster computing and other distributed computing environments. Applications in the field of particle physics, climate modelling, weather forecasting, astrophysics and EDA industry require vast compute resources generating massive amount of data[1]. These applications make use of high-end computers with large-scale data storage-systems that produce large multi-dimensional data sets. Distributed storage creates an overhead when compared to the traditional compute where local storage is tightly coupled with the system. Different architectures have been proposed [2, 3] to improve reliability, availability, scalability and reduce the downtime in a distributed environment.

Network based storage systems such as Network Attached Storage (NAS) [4] and Storage Area Network (SAN) [5] offer a robust and easy method to control and a access large amount of storage. Our work is based on NAS but the same can be extended to SAN as there is no factor, which limits the proposed model from being used with a different topology. Performance prediction of a storage system at a logical level, directly visible to users/schedulers generating the data is helpful for both the administrators and the users/schedulers. Informed action to control the future I/O intensive workload can be taken by the stakeholders. Whenever I/O intensive job is launched, the storage system tends to slow down in its response and when multiple such jobs are launched on the same storage system, unresponsiveness is encountered. In such scenarios, any further load (I/O intensive jobs launched) will lead to longer response time and reduction in the overall throughput of the entire compute environment. Hence, an intelligent framework capable of forecasting such high response periods will be of great importance for efficient utilization of storage as a resource, which in turn can enhance the throughput of compute environment.

The next section introduces the past literature, Section 3 talks about the storage architecture, which may vary for different vendors but gives a broad picture of the hardware components. Section 4 outlines the experimental setup used to develop the framework proposed in Section 5. Results and Conclusions are available in Section 6 and 7 respectively.

II. Literature

Predicting response time [6, 7, 8] or any performance parameter at the disk level is finely granular and due to advances in storage technology, disk level prediction may not add any value. There has been very limited research to have an automated mechanism to predict and alert the administrators or users/schedulers that can prevent the storage system becoming unresponsive. Response time prediction at a single disk level and Redundant Array of inexpensive Disks (RAID) level is researched in [7] which uses machine learning techniques. Although the prediction is carried out on standard workload traces, simulators are used to simulate disks on which workload traces are used to monitor the features. Influence of storage architecture and configuration used by the administrators are not considered in the list of features that are selected, except for the analysis of cache effect. Findings of [7] offer minimal view of the entire storage architecture with specialized hardware. Since analysis is at a disk and RAID level, the model cannot be extended to larger, more advanced storage systems that are used in [1].

Framework in [9] is an end to end root cause analysis, which supports abnormality detection What…if analysis carried out in [6, 8] answers questions related to distributed storage architecture. [8] talks extensively about various modules and self-prediction support in a cluster-storage architecture. This can accurately answer What..if type of questions about the performance impact of architectural and software changes in storage nodes. Response time prediction module predicts the response time as a function of throughput and the number of requests from the clients. This may not always be valid or accurate as the requests generated from the clients may or may not be generating I/O operations on a single logical area. When requests are distributed across various logical areas, the throughput prediction and in turn the response time prediction can be inaccurate. None of the past literature in [7, 8, 9] consider the complex aspects of a storage system rather focuses on the read/ write operations, latency, throughput and number of requests.

Ensemble analytical methods are used in [10] to predict the latency, but the ranking of the algorithms is not always optimal. The algorithms are a combination of analytic and machine learning models. One of the algorithm is chosen based on its accuracy level, which makes the other algorithms obsolete. Other storage related prediction involves disk drive failure that has been solved using either analytic or machine learning models [11, 12, 13]. Unified Storage architecture is becoming the industry standard and requires the prediction models to accommodate various specialised hardware dependent parameters and protocol specific parameters, which have not been considered in the past literature.

Forecasting the trend of a time series data was pioneered by BOX and Jenkins. The forecasting models are well established and utilized widely in various domains [14]. Extensive research has been carried out for predicting load in distributed environments [15]. Load prediction as an idea was first used in the electric GRID for power distribution [16]. This was later extended for short term CPU load prediction and load prediction in distributed systems. A much more interesting application is the runtime prediction of tasks in a distributed computing environment. Autoregressive integrated moving average (ARIMA) is one of the popular forecasting models used to forecast the values for storage parameters in the proposed work. It has been established that ARIMA works better for time series data when compared to Artificial Neural Networks[17]

We propose a framework consisting of an Ensemble of machine learning algorithms. This framework is robust and adapts to the workload that exists on the storage system. Statistical measures like Kappa coefficient[18, 19] and Sensitivity are used to rank individual models and use it for final classification. Forecast of future data points through ARIMA modelling is fed to the Ensemble classification model capable of identifying high response periods. With this proposed model, 95% of the test instances are classified accurately.

III. Storage Architecture

A storage filer is a specialized file server capable of handling high-volume data as a storage, backup, and archiving area. A typical storage architecture consists of multiple autonomous components for each filer. Multiple filers are grouped together in a cluster to fulfil the storage needs of the users. Various vendors provide a variety of specialized hardware to speed up I/O process. Performance of the filers vary based on the hardware capabilities that are available with each filer. The test filer used in the experiment consists of the following components:

- Physical Memory is a storage system memory for high speed processing.
- Non-volatile Random Access Memory (NVRAM) is a backup for system RAM in case of power failure or improper shutdown.
- Network Interface for users to communicate with the storage server.
- Protocol Stack supporting a variety of protocols namely - Common Internet File System (CIFS), Network File System (NFS), Fibre Channel (FC), Internet Small Computer System Interface (iSCSI), File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP).
- Write Anywhere File Layout (WAFL): Specialized file system to optimize write performance.
- RAID Layer supporting RAID 4 and RAID-DP (double parity).
- Disk array to store the data.

Each filer can be subdivided into logical layers that are vendor specific. Storage vendors provide multiple levels of logical separation. One such logical layer visible to the administrators is volume in NetApp or shares in Oracle Fishworks. Approximate illustration of the architecture is shown in Fig 1. Any further reference to a storage system in this paper will be with respect to a filer. Volumes are accessible through various protocols from the protocol stack layer, as configured by the administrator. Creating logical volumes allows the administrator to allocate storage as a resource to different entities in a large grid oriented industrial setup. This is the closest layer to the storage system that is viewable by

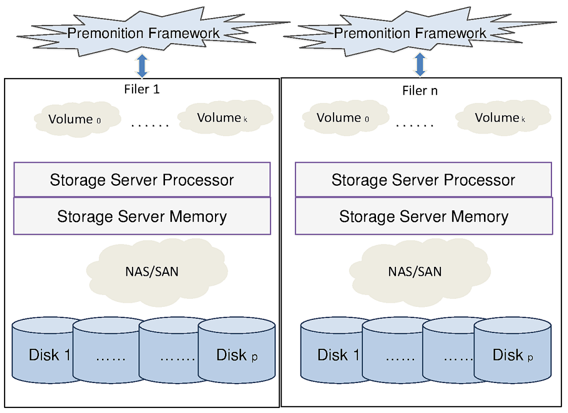

Figure 1. Architecture of a Filer

the administrators, they will be responsible for taking any action required when the response time is too high. Irrespective of the vendor, only one logical layer needs to be monitored. Rest of the logical layers can be eliminated since it will either be an aggregation of lower layers or a subset of the upper layer.

Various parameters related to the above mentioned components can be monitored. The detailed list of parameters are available in [20]. Out of the available 1000+ parameters that can be monitored in non-diagnostic mode, different workloads will affect different subset of the parameters. The configuration of individual filer at both physical and logical levels affect the parameter values. Every aspect needs to be accommodated to predict and classify with high accuracy. Different actions can be taken on the system side to prevent the unresponsive state using the proposed framework. We discuss some actions in detail in the Action Module in Section 5. The model will be vendor agnostic, and will adapt according to the type of workload that is being generated on the individual filer.

IV. Experimental Setup

The test filer used is a NetApp ONTAP 8 filer with Dual 2.8 GHz Intel P4 Xeon MP processor , 2 MB L3 cache, 8 GB of main memory and 512 MB NVRAM. A typical workload can generate a variety of I/O operations and to develop and validate the proposed model every possible I/O operation needs to be generated and tested. IOZONE, a popular benchmarking tool to generate synthetic data on a large scale was used to load the filer. It has the ability to generate 13 different types of I/O operations [21]. It's also necessary to simulate small-file operations similar to load encountered in web and mail servers. Response time can be measured for different file operations. Write operation is the costliest I/O operation. Hence, we use the time taken to complete a benchmark write operation as a measure of respons time.

Multiple instances of IOZONE and millions of small file operations were used to generate large I/O operations, with each instance generating different type of I/O operation at a time. During the process of loading the filer, response time for writing a file of size 500 MB is repeatedly carried out. I/O workload can follow any pattern and an effort was made to simulate the same. Scenarios with simultaneous large file write/read (200 GB each), combination of millions of metadata operations with large file write/read and sequential large file write/read were simulated. The experiment was carried out for a period of three days and the response of the filer was measured.
Filer parameters are collected through a multiprocess program with two processes being created. A parent process will launch a 500MB file write operation and busy waits for the child process to respond. Till the child process responds it will periodically collect all the filer parameters. When the child process responds , each vector of filer parameters collected will be assigned the same response class as the entire time period taken for the child process to respond. This helps in analyzing the general trend of the storage parameters for a particular response time.
Example : nfsv3_ops:306/s, net_data_recv:1222 KB/s, vol1_avg_latency:200μs, vol2_avg_latency:21μs, vol1_total_ops:2200KB/s, vol2_total_ops:2300KB/s , ..., response_time : 36 sec

The test setup was able to generate a workload with response times ranging from 6 seconds to 79 seconds for writing a file of size 500 MB. File listing was also considered as a possible response measure. File listing is a metadata operation that is not commonly used to test filer load. The administrator defines a threshold based on the architecture of the environment. This will be used as a cut-off for classifying the high response time. Class labels for each instance is appended based on the threshold. Aggregation of virtual and logical components of a filer is done as part of the preprocessing step in the proposed framework.

Our system will classify the response times at this logical layer for effective action to be taken either by the administrator or schedulers or the user. Let $V_{i\,j}$ denote the i^{th} instance of the j^{th}parameter. The parameters related to various volumes are available in each of the i^{th}instance along with other generic system parameters. Based on which volume is loaded, the volume parameters like the volume level read/write operations, read/write latencies, etc will vary. Let $V_{i\,sys}$ represent all monitored system parameters without any volume parameters for the i^{th}instance. We separate out the subset of volume parameters.

$$V_{i\,sys} = V_i - (V_{i\,vol_1} \cup \ldots \cup V_{i\,vol_k})$$

where vol_k denotes all the parameters related to k^{th}volumes This essentially helps us to predict the response time at individual volume vol_k by appending V_{vol_k} to V_{sys}.

Thus training set will effectively be

$$Training = V_{sys} + V_{vol_k}$$

Training a single volume on each filer will be enough to model the entire filer since the volume characteristics will not vary much on the same hardware. If we have a filer configured with two logical volumes, i.e V_1 and V_2,

$$V_{sys} = V - (V_{vol_1} \cup V_{vol_2})$$

With this setup we develop a machine learning Ensemble model where we use a combination of machine learning algorithms. Each algorithm has the capability of capturing linear and non linear patterns. We show that the Ensemble of the algorithms performs better than the individual algorithms.

V. PROPOSED FRAMEWORK

A. *Ensemble Methods*

Ensemble methods in machine learning try to combine various characteristics of a set of algorithms and predict the outcome for a new input data using the weighted technique or a voting mechanism. One of the most popular Ensemble methods is the Random Forest technique [22]. Random forest is a combination of multiple individual trees built by independently sampling the dataset with random selection of features. Having multiple trees predicting(voting) the class for each input vector and choosing the class which got the maximum votes has been shown to be very efficient in capturing relationship that exists in the data. It was shown that the generalization error which depends on strength of the individual trees tends to reduce as the number of trees in the forest grows. Random forest is known to be robust to noise and hence a preferred choice in our current framework. Other machine learning algorithms like Support Vector Machines (SVM) [23] and Self Organising Maps (SOM) [24] are known to effectively capture the non-linear mapping of the data which is essential for the current problem. Ensemble methods depend significantly on how to combine the outputs from multiple classifiers, to take advantage of individual classifiers which in turn improves the overall accuracy of the model. We use Skyline query [25] which is popularly used in economics and Database Management Systems to rank data points. We consider Sensitivity and Kappa coefficient as the dimensions for each model. Skyline query returns the rank based on these two parameters such that any higher ranked model always dominates all lower ranked model in all dimensions

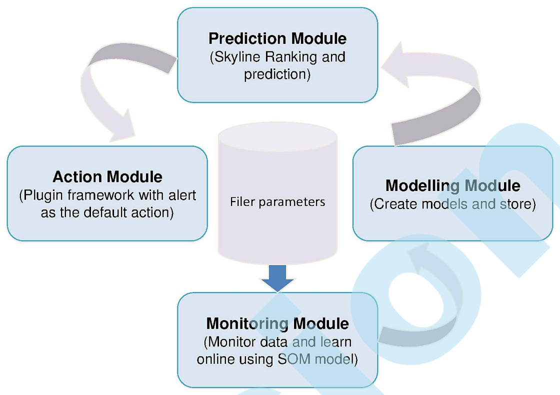
Prediction Module
(Skyline Ranking and prediction)
Action Module
(Plugin framework with alert as the default action)
Filer parameters
Modelling Module
(Create models and store)
Monitoring Module
(Monitor data and learn online using SOM model)

Figure 2. Proposed Framework

B. *Framework*

The proposed framework shown in Fig 2 consists of the following components:

1) *Monitoring module:* Continuously monitor the filer parameters and the response times, when the model needs to be generated for the first time. Monitoring module will record the response time till enough data has been collected. Typically data is collected for one full day which is the model refresh rate in the proposed solution.

Two stage of modelling needs to be carried out-

- ARIMA modelling by treating the storage parameters as a time series data. This model is build once in a day based on the entire dataset collected. Model is stored for prediction of filer parameters for the next n time period at each prediction interval.
- Modelling of Ensemble model is also carried out once in a day to adapt to the changes in workload if any and this will be used to classify the parameters forecasted by ARIMA into high response / low response class.

The model which is constructed will have all the non-volume parameters and the parameters of the volume which was used to train the model. Whenever the prediction has to identify the volumes which are experiencing high response times,

$$V_{sys} + V_{vol_k}$$

is used as the test vector. If the classification indicates high response time, then action can be taken indicating that vol_k will experience high response time and if preventive action is not taken, filer might become unresponsive.

Online learning of SOM [26] continues to learn with every new instance that is generated allowing SOM to adapt itself to the current workload. Features selection is a critical task in any machine learning algorithm. We use only those filer parameters which are highly correlated with the response time. A *correlation_threshold* of +/- 0.1 is used to select features. All the features which have correlation greater than +0.1 or lesser than -0.1 are considered. Once these features are selected they will be used for the current filer as long as the workload pattern doesn't change significantly as measured by the SOM model accuracy. The process of monitoring and SOM learning continues till the prediction accuracy of the SOM model achieves the desired threshold, i.e. *SOM_threshold* defined by the administrator.

SOM is chosen for initial model development as it adapts to the changing behaviour of the workload and is able to stabilize irrespective of the disparity in the number of instances that belongs to either classes. Once the SOM model achieves the desired accuracy, it initiates the Modelling module. When there is a significant drop in the prediction accuracy of SOM model, the latest monitored dataset will be used for regenerating the new model. Maintaining the history is necessary to adapt and recreate an Ensemble model in order to improve prediction accuracy when workload changes.

2) Modelling module: A Skyline ranked (Pareto Optimal) Ensemble model is developed which builds three individual models *viz* SOM, Random Forest and SVM. The Random Forest algorithm is well known for generalizing for different classes of datasets. Simultaneously SVM is built which will try to capture the non-linear relationship of the dataset. Based on past literature, it's been found that SVM is one of the better performance classifiers for storage for limited application dependent parameters. Not all possible patterns can be captured individually by a single model. Hence, we propose an Ensemble of these algorithms which will consider all the three models and rank them using Skyline queries. This helps to rank the models based on multiple statistical parameters. Accuracy of each model is cross-validated individually and the system is deployed only if the accuracy of all the models is above the *deployment_threshold* which can be similar to *SOM_threshold*.

3) Prediction module: Prediction module will run the predict for each available volume and identify the list of volumes in the filer which belongs to a class of high response time. When it comes to predicting high response class which will eventually lead to unresponsiveness due to timeout set at the OS level, we are concerned about having a model which is able to predict the positive class(volume has high response time), measured by HIGH Sensitivity of the model. Sensitivity of a model is defined as the rate of classifying a Positive class accurately. This is also referred to as the True Positive Rate.

$$Sensitivity = TruePositives/Positives$$

Models with high Sensitivity are preferred in the current scenario as we cannot afford to miss a high response class being classified as a low response class. Hence we use the Skyline query to rank the three models based on their Sensitivity and Cohen Kappa coefficients. Cohen's Kappa coefficient is popularly used in statistics which is a measure of the inter rater aggrement. This has also been used as a performance measure for feature selection in [19]. Kappa coefficient is measured as

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)},$$

where $P(A)$ is the rate of agreement by all raters, and $P(E)$ is the rate of agreement by chance by all raters.

Based on the ranking of the model using Sensitivity and Kappa coefficient, we give importance to the top two ranked models and the third ranked model is used to strengthen the classification. Hence the rank of the individual models help the Prediction system decide which two models are to be considered for obtaining the classification result. The modelling module constantly updates the rank of each model with every tested instance.

Prediction module consists of two steps.

- A short term filer parameters prediction followed by classification of the predicted parameters.
- Skyline ranked Ensemble model is applied on the predicted parameters, the following rules are used:
  - *Rule 1*: If all 3 models predict class A (3 votes) $\Rightarrow$ Prediction of the system is Class A
  - *Rule 2*: If top ranked model predicts class A with at least one more model predicting the same (2 votes) $\Rightarrow$ Prediction of the system is Class A irrespective of the prediction by other model.
  - *Rule 3*: If top ranked model predict class A (1 vote) and the rest of the models predict Class B (2 votes) $\Rightarrow$ Prediction of the system is will be based on difference in Sensitivity of top 2 ranked models. Sensitivity should not differ by more than *sensitivity_threshold*.
    - *Rule 3a*: If the difference in Sensitivity is greater than *sensitivity_threshold*, $\Rightarrow$ Class A is predicted.
    - *Rule 3b*: Else $\Rightarrow$ Class B is predicted as it has acquired 2 votes.

The prediction module lists the volumes that are experiencing or tending towards high response time. This list will be processed by the action module.

4) Action module: Action module is an unimplemented module which can be used to take variety of decisions.

- Report to the administrator with the list of volumes which are in a critical level experiencing high response time or tending towards high response time if the I/O is not controlled.
- Report the user about the possibility of high response time in the near future and expect the users to act sensibly.
- Communicate to the scheduler to stop further requests for *'n'* time intervals on these volumes.
- Can be configured to kill any one task/job which is generating load to get the system to a responsive state.

Here we can draw an analogy with the deadlock situation of an Operating system. Deadlock recovery algorithms demand killing some of the processes which are involved in a deadlock scenario. Similarly in a Grid computing environment, multiple I/O intensive jobs generating vast amount of I/O operations lead to a high response time. Classical OS approach is followed by storage administrators in a large Grid environment where jobs are suspend to recover an unresponsive filer. Prevention mechanisms are a must to maintain high performance, efficiency and reliable delivery of storage as a resource.

What else can be done? Apart from other benefits already mentioned, this framework can be used by vendors to

- Identify the parameters (features with high correlation with response time) which are forming the bottleneck for slower response times for specific workloads. This can help the vendors to address the bottleneck issues from the identified parameters and provide specialised industry specific hardware/software solutions to improve the performance of their storage solution.
- This can also help the administrators to identify the parameters which can be configured at their end to provide better performance.

As shown in Fig 1, each individual filer will have its own proposed framework and will learn based on the workloads generated on the filer. It's very common for different types of jobs generate different types of workloads and the load on the filer depends on its hardware configurations. Each filer's performance is vendor dependent and having a generalized model is difficult which can comprise with the accuracy of the model. Hence each filer has its own Premonition framework and features (prediction parameters) are extracted separately for each filer which will learn and independently predict the response class. The entire framework was implemented using R Statistical Computing package [27]. We used R Perl APIs to connect the monitoring module with the Modelling and Prediction module.

A summary of the steps followed in the proposed framework is listed below:

- Every model is refreshed on a daily basis to accommodate any change in workload.
- Everyday a set of storage parameters like read, write , latency of various components, volumes along with hardware, network and protocol specific parameters like nvram , ext_cache, cifs_ops, nfs_ops, ifnet etc... [20] are monitored.
- As the data is collected, SOM is continuously trained with the data collected and its accuracy is monitored constantly.
- Model accuracy is evaluated to see if it is better than *SOM_threshold*.
- If the system has learnt sufficiently beyond the *SOM_threshold*, ARIMA modelling is carried out for each parameter individually to identify the most accurate value of *'p'*(which determines the number of past instances to be considered for future forecasting). This is achieved by identifying the value of *'p'* for which Akaike's information criterion (AIC) is minimum and 90% Confidence Interval (CI) of the fitted model is maximum.
- After identifying and saving the ARIMA model, Random Forest and SVM models are built simultaneously with the available dataset being divided into training and cross validation dataset.
- Based on the accuracy of the built models, the Ensemble model (ENS_{new}) is either accepted or rejected.
- If ENS_{new} is accepted for deployment, then
  - ARIMA will start predicting the individual parameters for next 15 mins (*'x'* intervals where *'x'* = (15*60)/*data_collection_frequency*).
  - Based on the predicted parameters, Ensemble model will predict the response time classification for all *'k'* volumes.
  - If any of the *'k'* volumes is predicted to fall in the high response time category, then the Action module will take one of the many possible actions as configured.
- If ENS_{new} is rejected for deployment, then
  - A subset of the training dataset is used as test dataset to the previous model (ENS_{old}) which was successfully deployed.
  - If the ENS_{old} meets the *deployment_threshold* criteria, then it will be retained and deployed.
  - Else

    $$max(accuracy(ENS_{old}), accuracy(ENS_{new}))$$

    is chosen as the model to be deployed.

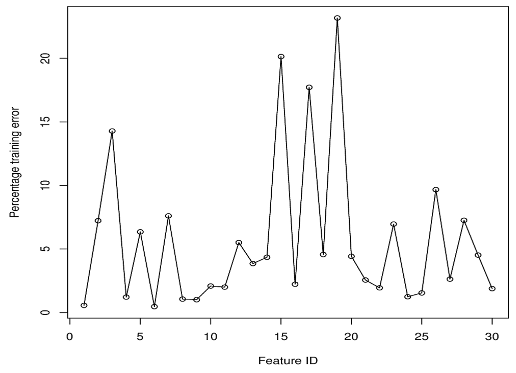

Figure 3. ARIMA Error rate for Training dataset

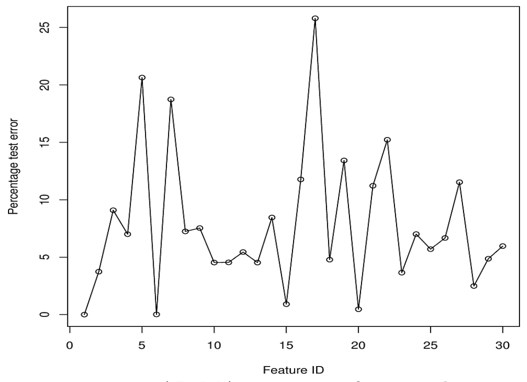

Figure 4. ARIMA Error rate for Test dataset

VI. Results

The experimental setup generated a dataset where response times were recorded for basic operations like listing files and creating a file on volumes. The response times for writing 500 MB of data were ranging from 6 seconds (no load) to 79 seconds. 24,146 instances were collected of which 70% was used in the training phase and 30% of the instances were used for validating the model. Based on the response times obtained, the dataset was split into two classes namely "High Response class" and "Low Response class". Sensitivity and Kappa coefficients are calculated with respect High Response class which is defined as the positive class. The dataset has some inclination towards high response time instances which may be a typical scenario which will be of interest to administrators and users. An *SOM_threshold* of 90% and *sensitivity_threshold* of 0.1 was used in the experiment. 60 seconds of response time is the cut-off for classification for high response time. All responses which took more than 60 seconds are labelled as high response class. SVM with Radial kernel function and Random Forest model consisting of 500 trees were built. SOM was modelled in a *4x4* grid which was optimal in-terms of speed and accuracy.

ARIMA model which will predict the parameters for next *'n'* intervals needs to be accurate for the Ensemble classification model to be useful. The normalised percentage error for the ARIMA model which fits the training dataset is shown in Fig 3. It is evident that the percentage error of the ARIMA model compared to the actual data is very low for various parameters with errors ranging from 0.6% for individual volume latency to 23% for ifnet data received. The value of *'p'*, i.e Auto Regression coefficient was identified to be between 9 and 15. All the filer parameters are predicted independently using *'p'* specific to each filer parameter. Fig 4 shows the percentage test error to predict filer parameters which is similar to the error observed in the fitted model in Fig 3. Since we have only two classes and the range of values for individual filer parameters are very high (order of 10^6 for most parameters), the tolerance level to this error is also high. Hence errors of 10 to 15% doesn't affect the classification accuracy to a large extent.

Table I shows the importance of each of the 30 features (measured by GINI index in Random Forest) that were used for modelling the system. It should be noted that these features will change with variation in workload. Different filers and different vendors can come across multiple such subsets of features.

Table I
Importance of features in Random Forest

Feature	GINI Index
volume_total_ops	194.2
volume_read_latency	183.8
volume_other_ops	169.6
volume_other_latency	130.6
volume_write_data	123.2
system_disk_data_read	121.6
system_net_data_sent	116.0
volume_write_ops	111.7
nfsv3_read_latency	109.9
vfiler_net_data_sent	109.7
ifnet_send_data	98.2
vfiler_read_bytes	96.5
system_disk_data_written	87.3
system_net_data_recv	86.4
ifnet_recv_data	83.2
nfsv3_write_ops	82.6
system_cpu_busy	79.2
vfiler_net_data_recv	78.8
vfiler_write_bytes	78.3
ifnet_send_packets	76.8
system_cpu_busy	73.1
vfiler_write_ops	73.0
vfiler_read_ops	66.3
vfiler_cpu_busy	63.4
nfsv3_read_ops	62.8
system_total_processor_busy	52.3
system_nfs_ops	26.3
system_avg_processor_busy	24.0
nfsv3_write_latency	22.8
nfsv3_ops	22.1

Proposed framework is tested with two different datasets generated by IOZONE.

- Dataset 1 consists of a fixed pattern with low followed by high I/O operations that are generated for a period of time. This dataset consists of 2,034 instances of which 1,017 had High response time and 1,017 instances had low response time.

- Dataset 2 consists of a 9,278 instances of which 3,408 instances had low response times and 5,870 instances had high response time. This dataset consists of a fixed pattern of 2,000 instances followed by a 7,278 instances of small bursts of random I/O operations. These operations generate high momentary I/O operations which if persistent will lead to unresponsiveness.

Dataset 1 is a typical scenario where I/O operations rise and fall based on the workload and follows a pattern. Dataset 2 is used to simulate a scenario where there are workloads which generate high I/O for shorter duration but will not persist with the same pattern for longer duration.

The test results for individual models are shown in Table II. The proposed Skyline Ranked Ensemble Model outperforms other models in all aspects i.e. accuracy, Sensitivity and Kappa coefficient. Proposed model was able to achieve a success rate of 98.3% and 95.4% for Dataset 1 and Dataset 2 respectively. Fig 5 shows the plot for success rate for Dataset 1. It is evident from the plot that initially SVM and SOM have very high classification accuracy. This trend changes drastically at around mid point. Similar change is not seen in Random Forest which stabilises and predicts the classes accurately throughout. Initially, the proposed model adapts to SVM / SOM and at a later stage it adapts to Random Forest whose rank increases with more test instances being used. Fig 5 shows the advantage of ranking based on statistical measures which helps the ensemble model adapt to the best algorithms which are capable of accurately modelling the varying workload pattern.

Success measure for Dataset 2 is shown in Fig 6. The trend is similar to results obtained from Dataset 1, but both SVM and SOM are able to adapt the the workload over time. As the Sensitivity and Kappa coefficient of SVM and SOM improve, they start contributing in the voting mechanism. Instances where Sensitivities are same for more than one model, Kappa coefficient helps to rank the models appropriately since it considers the chance agreement while classifying. Dataset 2 shows the fact that over a longer period of time where workload is random in nature, all three models perform well but the proposed Ensemble model has the best accuracy. It's able to get better success rate than the three models or atleast perform as well as the best model available (Random Forest in this case).

Out of 9,278 instances in Dataset 2, voting mechanism (Rule 1, Rule 2 or Rule 3b) was used for classifying 1,080 instances and all 1,080 instances were classified accurately. Rest of the 8,918 instances were determined using Rule 3a with a *sensitivity_threshold* of 0.1. The involvement of different algorithms in ranking for Dataset

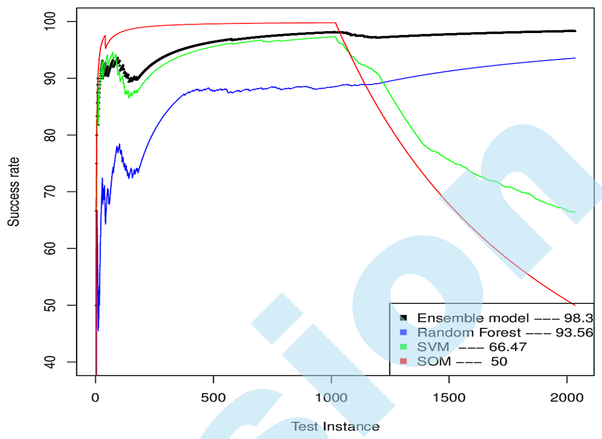

Figure 5. Success rate for all models on Dataset-1

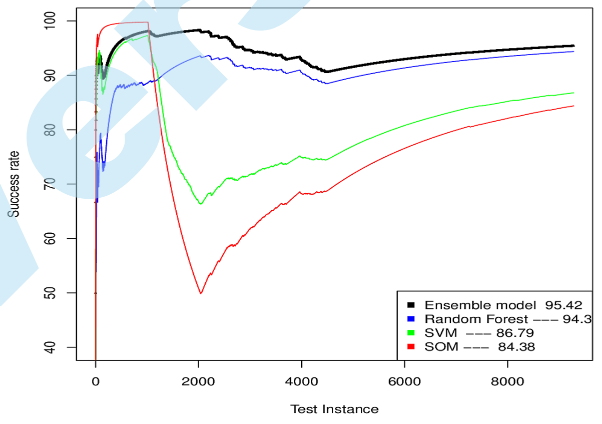

Figure 6. Success rate plot for all models on Dataset-2

2 is shown in Fig 7. If we had to use one of the three models to classify, then the preferred choice would have been Random Forest because of its high overall accuracy. But its clearly visible in Fig 6 that Random Forest takes a while to start classifying accurately. Hence ranking of models is necessary and more emphasis is stressed on picking the model which is accurately predicting the positive class (i.e. High response class).

One important parameter that needs to be set is the *sensitivity_threshold*. This was set to 0.1 in our experiment. If this is set to a higher value, the number of positive predictions can increase since more instances follow Rule 1/Rule 2/Rule 3b. The number of false alarms may also increase in such scenarios since the *sensitivity_threshold* is effectively setting the tolerance level for considering multiple models. Fig 8 shows the involvement of different algorithms in ranking when the *sensitivity_threshold* is set to 0.3. 2,329 instaces were involved in voting of which 2,316 were classified accurately. Increasing the threshold reduced the success rate to 95.28% with increase in false alarms from 382 (for *sensitivity_threshold* of 0.1) to 395 instances.

Table II
Performance Analysis

Dataset	Model	Success Rate	Sensitivity	Kappa coefficient
Dataset 1	Random Forest	93.6	0.984	0.871
	SVM	66.5	0.602	0.329
	SOM	50.0	0.500	0.1
	Skyline Ranked Ensemble Model	**98.3**	**0.985**	**0.967**
Dataset 2	Random Forest	94.3	0.938	0.877
	SVM	86.8	0.844	0.699
	SOM	84.4	0.632	0.803
	Skyline Ranked Ensemble Model	**95.4**	**0.938**	**0.899**

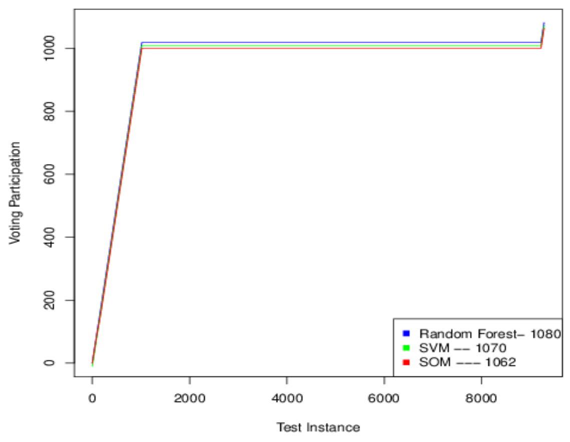

Figure 7. Number of Classifications using voting mechanism for Dataset-2 with *sensitivity_threshold* of 0.1

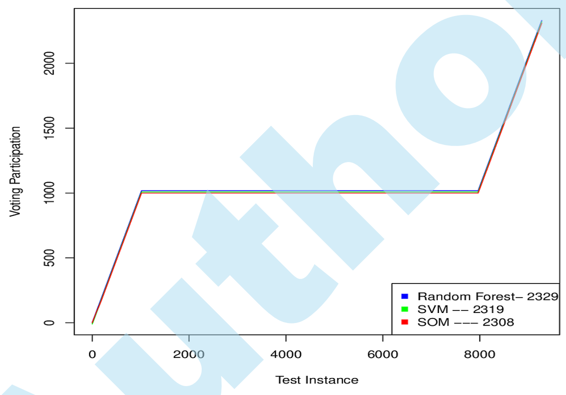

Figure 8. Number of Classifications using voting mechanism for Dataset-2 with *sensitivity_threshold* of 0.3

VII. Conclusion

Skyline query mechanism helps to strengthen the classification by ranking the models based on multiple criteria. This forces the weaker models to go down the ranking over time which in turn prevents misclassification. Practical implementation of the proposed model will improve the compute efficiency significantly, especially in environments with a variety of vendors and topologies. The setup works irrespective of the vendor and they type of workload that is being generated. This is because of the following facts:

- Correlation of the filer parameters with the response time are chosen as features for training the model. This makes the models vendor agnostic but adding an additional overhead of having separate model for each independent filer available in the environment.
- Each filer will have its own separate model as the range of parameters will vary based on vendor. The values will vary based on the workload generated. Hence separate models to adapt to each different workload helps in creating a more effective prediction model.
- The model is refreshed every day to accommodate any change in the workload making the entire system self adapting.

It should be noted that Random Forest alone can be used if the type of scientific workload that will be generated is going to be consistent on a filer.

Future work will include better statistical parameters (e.g. Specificity) to deduce more statistical information to improve classification. The need to maintain historical data can be eliminated by making all the models learn online. More accurate forecasting of filer parameters might be necessary, as inaccurate forecast will lead to misclassification in the transition period from low response time to high response time and vice-versa. Forecasting response times using regression analysis instead of forecasting the filer parameters is an option to reduce misclassification. Testing the proposed model with more datasets and showing the statistical significance of the proposed model using Friedman's test should done.

References

[1] Network Appliance, Inc. (2012) CERN: Unlocking the secrets of the universe. Accessed: 2014-03-30. [Online]. Available: http://www.netapp.com/us/media/ds-3365-0812.pdf

[2] X. Shen and A. Choudhary, "A distributed multi-storage resource architecture and IO performance prediction for scientific computing," in *High-Performance Distributed Computing, 2000. Proceedings.*, 2000, pp. 21–30.

[3] J. Zhang and P. Honeyman, “A replicated file system for grid computing,” *Concurrency and Computation: Practice and Experience*, vol. 20, no. 9, pp. 1113–1130, 2008.

[4] Y. Deng, “Deconstructing network attached storage systems,” *Journal of Network and Computer Applications*, vol. 32, no. 5, pp. 1064 – 1072, 2009.

[5] A. Shoshani, A. Sim, and J. Gu, “Storage Resource Managers: Middleware Components for Grid Storage,” 2002.

[6] D. Narayanan, E. Thereska, and A. Ailamaki, “Continuous resource monitoring for self-predicting dbms,” in *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, ser. MASCOTS ’05, 2005, pp. 239–248.

[7] C. Dai, G. Liu, L. Zhang, and E. Chen, “Storage device performance prediction with hybrid regression models,” in *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2012 13th International Conference on*, Dec 2012, pp. 556–559.

[8] E. Thereska, M. A. el malek, J. J. Wylie, D. Narayanan, and G. R. Ganger, “Informed data distribution selection in a self-predicting storage system,” in *In International Conference on Autonomic Computing*, 2006, pp. 187–198.

[9] K. Pollack and S. Uttamchandani, “Genesis: A scalable self-evolving performance management framework for storage systems,” in *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, 2006, pp. 33–33.

[10] J. Chen, G. Soundararajan, S. Ghanbari, and C. Amza, “Model ensemble tools for self-management in data centers.” in *ICDE Workshops*, 2013, pp. 36–43.

[11] G. Hamerly and C. Elkan, “Bayesian approaches to failure prediction for disk drives,” in *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 2001, pp. 202–209.

[12] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma, “Proactive drive failure prediction for large scale storage systems,” in *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on*, May 2013.

[13] E. Pinheiro, W.-D. Weber, and L. A. Barroso, “Failure trends in a large disk drive population,” in *Proceedings of the 5th USENIX Conference on File and Storage Technologies*, ser. FAST ’07, 2007, pp. 2–2.

[14] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[15] P. A. Dinda and D. R. O’Hallaron, “Host load prediction using linear models,” *Cluster Computing*, vol. 3, no. 4, pp. 265–280, Oct. 2000.

[16] R. Li, J. H. Li, and H. M. Li, “The short-term electric load forecasting grid model based on mdrbr algorithm,” in *Power Engineering Society General Meeting, 2006. IEEE*, 2006.

[17] J. C. Palomares-Salas, J.-J. de la Rosa, J. G. Ramiro, J. Melgar, A. Aguera, and A. Moreno, “Arima vs. neural networks for wind speed forecasting,” in *Computational Intelligence for Measurement Systems and Applications, 2009. CIMSA ’09. IEEE International Conference on*, May 2009, pp. 129–133.

[18] J. Cohen, “A Coefficient of Agreement for Nominal Scales,” *Educational and Psychological Measurement*, vol. 20, no. 1, p. 37, 1960.

[19] S. Vieira, U. Kaymak, and J. M. C. Sousa, “Cohen’s kappa coefficient as a performance measure for feature selection,” in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, July 2010.

[20] Network Appliance, Inc. (2010) Netapp unified storage performance management using open interfaces. Accessed: 2014-03-30. [Online]. Available: https://communities.netapp.com/servlet/JiveServlet/previewBody/1044-102-2-7517/Performance_Management_DesignGuide.pdf

[21] William D. Norcott, Don Capps. (1998) Iozone filesystem benchmark. Accessed: 2014-03-30. [Online]. Available: www.iozone.org/docs/IOzone_msword_98.pdf

[22] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[23] J. A. K. Suykens, “Nonlinear modelling and support vector machines,” in *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*, vol. 1, May 2001, pp. 287–294 vol.1.

[24] T. Kohonen, “The self-organizing map,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep 1990.

[25] X. Liu, D.-N. Yang, M. Ye, and W.-C. Lee, “U-skyline: A new skyline query for uncertain databases,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 4, pp. 945–960, April 2013.

[26] J. Feng, C. Zhang, and P. Hao, “Online learning with self-organizing maps for anomaly detection in crowd scenes,” in *Pattern Recognition (ICPR), 2010 20th International Conference on*, Aug 2010, pp. 3599–3602.

[27] R. D. C. Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2010.