A Profile Guided Approach to Scheduling in Cluster and Multi-cluster Systems

Arvind Sridhar and Dan Stanzione Ph. D.
{ahsridha, dstanzi}@asu.edu

Fulton High Performance Computing, Arizona State University

Abstract. Effective resource management remains a challenge for large scale cluster computing systems, as well as for clusters of clusters. Resource management involves the ability to monitor resource usage and enforce polices to manage available resources and provide the desired level of service. One of the difficulties in resource management is that users are notoriously inaccurate in predicting the resource requirements of their jobs. In this research, a novel concept called 'profile guided scheduling' is proposed. This work examines if the resource requirements of a given job in a cluster system can be predicted based on the past behavior of the user's submitted jobs. The scheduler can use this predicted value to get an estimate of the job's performance metrics prior to scheduling and thus make better scheduling decisions based on the predicted value. In particular, this approach is applied in a multi-cluster setting, where the scheduler must account for limited network bandwidth available between clusters. By having prior knowledge of a job's bandwidth requirements, the scheduler can make intelligent co-allocation decisions and avoid co-allocating jobs that consume high network bandwidth. This will mitigate the impact of limited network performance on co-allocated jobs, decreasing turnaround time and increasing system throughput.

1 Introduction

As the need for computational power increases, cluster computing has become a viable solution for compute intensive applications.The main challenge in cluster computing is resource management which involves the ability to monitor resource usage and enforce policies to manage the available resources and provide the desired service. Overestimation of resource requirements of a job leads to poor resource utilization. Underestimation of resource requirements leads to degradation of job's performance.

The single cluster systems are shared by large number of users, and hence proper scheduling and resource management are essential for good system performance and utilization. The resource management system is responsible for managing the system load and prevent jobs from competing with each other for compute resources [2]. The scheduler handles the memory allocation, queue management, resource reservation, minimizing queue times, turn-around times and lot of other important functions.

The benefits of cluster computing have given rise to Multi-cluster systems. As multi-clusters become increasingly prevalent, efficient resource allocation among the available resources becomes a challenging task. Upon further inspection, the multi-clusters are inter-connected via a dedicated network and hence inter-cluster network bandwidth and network utilization play an important role in

resource allocation and scheduling. The process of allocating resources across cluster boundaries for servicing job requests is called co-allocation. If jobs that are co-allocated, perform a large number of inter-task communications, the network bandwidth on the inter-cluster link is consumed. If the bandwidth usage of these jobs exceeds the bandwidth capacity of the inter-cluster link, the co-allocated jobs will experience slow down. This increases the turn-around time of the jobs and hence degrades the performance and throughput of the scheduler. These problems can be avoided, if the scheduler aware of jobs's bandwidth requirements. The scheduler can then avoid co-allocating jobs that consume high bandwidth. A possible solution is have the users provide the resource requirements. However, previous research has shown that user provided estimates are highly in-accurate for runtime [3] [4] and likely to be more so for memory, network bandwidth etc.

In this work we propose a novel concept called 'Profile guided scheduling'. We develop a predictor that forecasts the performance metrics of a job, based on the execution history of jobs submitted by a user. The scheduler can use the predicted values to get an estimate of job's performance metrics prior to scheduling and hence make better scheduling decisions. Since the performance of co-allocated jobs in multi-cluster systems is dependent on the availability of the inter-cluster network bandwidth, the scheduler can use the predicted bandwidth value to get estimate of job's bandwidth characteristics prior to scheduling and thus make better co-allocation decisions.

2 Related Work

2.1 Cluster Workload Characteristics

In cluster computing or grid computing, workloads typically contain a pool of users with different activity levels and periods [5]. Users tend to submit similar applications repeatedly in order to get a wide range of results. These applications are generally scientific applications that involve large number of computations or simulations, and are submitted to the cluster repeatedly by changing a few parameters, to obtain large quantity of results or simulation data.

Workload characterization of clusters and grids are very important to understand the system behavior, performance and develop workload models for evaluating different system designs and scheduling strategies. It is very useful to generate synthetic workloads which can be used for performance evaluation of scheduling algorithms.

Workload characterization also helps in prediction or forecasting performance metrics. This involves collecting historical workload data and applying statistical techniques to predict the metrics based on the collected workload data. The workload parameters that are generally of interest in cluster computing include memory consumed, job arrival rate, job size, execution time, CPU utilization, network bandwidth consumed, user or group behavior as these parameters have an influence on scheduling. The correlations between these metrics often give valuable information.

Analysis of workloads also helps in drawing relationships and dependencies among various metrics. For example, the workload analysis at Cornell Theory Center [6] shows that jobs requesting smaller number of processors had relatively shorter run times than those requesting large number of processors. Another interesting observation is the 'power of two phenomena' [5], where jobs submitted

by the user generally request for 'power of two' number of processors. In the workload characteristics of NASA Ames iPSC /860 [7], it is seen that some users tend to submit small jobs and run them repeatedly while others submit jobs overnight for long periods. When multiple executions of the same application and same number of nodes were considered, the coefficient of variation was found to be less than 1, indicating that historical information could be used to estimate the resource requirements. Another feature that is widely observed is that users whose application is under development request fewer number of processors when compared to users who submit fully functioning applications.

From the study of workload characteristics of clusters we find that user activity and characteristics of jobs submitted by a user have a strong correlation which can be exploited for scheduling. By having the execution history of jobs submitted by users or groups, important information can retrieved which can be used to improve scheduling of jobs in clusters and multi-clusters.

2.2 Prediction or Forecasting in Clusters

Study of cluster workloads gives an insight on the parameters or metrics for which prediction techniques can be applied to predict the future values. There are static metrics such as number of processors requested and nodelist which remain constant throughout the execution of the job. On the other hand are dynamic metrics like queue wait time, execution time, network utilization, CPU Utilization and many more. The dynamic metrics, if predicted accurately, can influence scheduling decisions.

The Network Weather Service (NWS) [8] project is a good example of forecasting in grid computing environment. The NWS is a distributed system providing short term performance forecasts of CPU load, memory and other metrics. The NWS uses monitors to gather the performance metrics. The NWS is mainly intended to provide resource performance forecasts in meta-computing environments. It uses mean, median and auto-regressive based prediction algorithms to forecast the metrics. The system tracks the accuracy (using prediction error) of all predictors, and uses the one exhibiting the lowest cumulative error measure at any given moment to generate a forecast.

Li [11] has proposed a system to predict job start time in clusters. In this approach 'templates of job characteristics' based on the statistical properties of workload traces are defined to identify similar jobs that have executed in the past. Two statistical estimators: Auto-regressive model and Linear regression are used for predictions. If the predictor with the smallest error cannot derive a prediction, a broader template is used until a valid prediction is made. Experimental results of this approach have shown that good accuracy in prediction of job start times.

Statistical techniques such as liner regression [10] are also widely used for prediction. Linear Regression analyzes the relationship between the dependent and independent variables for making predictions. Using this technique a regression equation is obtained in which the dependent variable is modeled as a function of independent variables, constant and an error term. Linear regression works well for predictions within the data present in the data set but does not work well for extrapolation.

Although there has been some work on prediction and forecasting of metrics in clusters, they focus on predicting the system load, job start time or resource performances. In [11] and [12] simulations aimed at predicting queue wait time

and job start time have been performed, but there are drawbacks in these approaches. The prediction of dynamic metrics like bandwidth utilization, memory usage of jobs, aimed to improve scheduling in clusters and co-allocation in multiclusters has not been explored much.

3 Design of Profile Guided Predictor

The developed predictor forecasts the performance metrics of a job based on the past behavior of jobs submitted by a user. In order to predict from the past behavior, an execution history for each user has to be maintained. In this research, we call this execution history as the User's profile. The User's profile information includes the execution time, network bandwidth, memory consumption and CPU Utilization of each job submitted by the user to the cluster. A database is used to store the user's profile. When a prediction is to be made, the required data is fetched from the user's profile and statistical algorithms are applied on this data to get the predicted performance metrics. The profile guided predictor predicts three performance metrics: Per Processor Network Bandwidth, Average Per Processor Memory and CPU Utilization.

3.1 Predictor Operations

We consider five different prediction algorithms to study how these algorithms perform on the different user's profiles. Each of these prediction algorithms behave differently depending on the performance metric data present in the user's profile. The five different algorithms are explained below:

History

This predictor uses the entire execution history of the user to predict the performance metrics. This includes all the metrics recorded since the first submission of a job by the user. The predicted metric is obtained by taking the mean of the entire execution history of a user.

Consider a user's profile containing an execution history of 'k' bandwidth's $\{BW_1, BW_2 \ldots BW_k\}$. The bandwidth predicted by the history predictor is given by:

$$BW_{predicted} = \sum_{i=1}^{k} BW_i/k, \tag{1}$$

In general, for any performance metric, the predicted value is given by:

$$Metric_{predicted} = \sum_{i=1}^{k} Metric_i/k, \tag{2}$$

Sliding Window

In this predictor, the amount of historical data used for prediction can be varied. Sliding window uses a limited history of user's profile to make predictions. The predictor returns the mean of the selected values as the predicted metric. The scheduler can vary the 'window size' and select a set of most recent jobs for prediction and discard the earlier ones.

For Example, Consider a user's profile containing an execution history of 'k' bandwidth's $\{BW_1, BW_2 \dots BW_k\}$. Using this predictor, we can use the last 'j' bandwidths recorded to predict the bandwidth.

$$BW_{predicted} = \sum_{i=j}^{k} BW_i/(k-j) \tag{3}$$

where $j < k$

In general, for any performance metric:

$$Metric_{predicted} = \sum_{i=j}^{k} Metric_i/(k-j) \tag{4}$$

where $j < k$

Weighted

The Weighted predictor tries to determine an interval, in which most of the user's jobs bandwidth or memory values exist. It then predicts a value close to this interval. This predictor assigns weights to each data value, such that values that are assigned with larger weights contribute more to the predicted value. The predictor divides the user profile data into intervals. The frequency distribution is determined which gives the number of values falling in each of the intervals. The interval with maximum frequency is given greatest weight. This predictor operation is useful when the performance metrics in the user's profile are distributed over a small range of values.

The predicted value is calculated using the following algorithm. Consider the performance metrics of 'j' jobs in a user's profile.
$Profile_{metric} = \{Metric_1, Metric_2 \dots Metric_j\}$.

Algorithm 1 Weighted Predictor Algorithm

Find the distribution of data in the users profile.
  Calculate $Metric_{max}$, $Metric_{min}$ from $Profile_{metric}$
  $Range = Metric_{max} - Metric_{min}$
  Divide *Range* into k Windows, $\{W_1, W_2 \dots W_k\}$.
  Calculate frequency of each Window.
  for each $Metric_j$, find the window W_k, such that $Metric_j \in W_k$.
    if $Metric_j \in W_k$ then increment $Frequency(W_k)$ by one.
  Calculate Weights for each Window.
    $Weight_{W_i} = Frequency(W_i)/j$ where $0 < i < k$
  for every $Metric_j$, find the Window W_k such that $Metric_j \in W_k$
    if $Metric_j \in W_k$ then $Metric_j = Metric_j \cdot W_k$

$$Metric_{predicted} = \sum_{i=1}^{j} Metric_i \cdot W_i / \sum_{i=1}^{j} W_i$$

The number of windows into which the data set is divided can be changed by varying 'k'. Greater the value of 'k', more is the precision of the predicted value.

Locality

A very common observation in cluster computing is that, users submit a job to the cluster, get the results and submit the same job again by just varying a few parameters like number of simulation runs or change the input file for the application. The Locality predictor tries to exploit this behavior of the users for predicting the performance metrics. This predictor returns the most recent value recorded in the user profile as the predicted value.

Consider a user's profile containing an execution history of 'k' bandwidth's $\{BW_1, BW_2 \dots BW_k\}$. The bandwidth predicted by the Locality predictor is given by:

$$BW_{predicted} = BW_k$$

For any performance metric, the predicted bandwidth is given by:

$$Metric_{predicted} = Metric_k$$

Exponential

The Exponential predictor follows the time series technique called exponential smoothing to predict the performance metrics. It is called 'Exponential Smoothing' because it assigns exponentially decreasing weights as the observations get older. In other words, recent observations are given relatively higher weight in forecasting than the older observations.

The basic exponential smoothing formula is given by:

$$S_t = \alpha Y_t + (1 - \alpha) S_{t-1} \tag{5}$$

where
Y is the original series
S_t is the singly smoothed series obtained by applying exponential smoothing to series Y
α is the 'smoothing constant' and $0 < \alpha < 1$

The single exponential smoothing does not work well when there are trends or sudden deviations in data. This situation is countered using the doubly-smoothed series. This is calculated by applying simple exponential smoothing to series S_t using the same α.

$$D_t = \alpha S_t + (1 - \alpha) D_{t-1} \tag{6}$$

where
Y is the original series
S_t is the singly smoothed series obtained by applying exponential smoothing to series Y
D_t is the double smoothed series obtained by applying exponential smoothing to series S
α is the 'smoothing constant' and $0 < \alpha < 1$

The predicted value is given by:

$$Y_{predicted} = a_t + b_t \tag{7}$$

where

$$a_t = 2S_t - D_t \tag{8}$$

$$b_t = (\alpha/(1-\alpha))(S_t - D_t) \tag{9}$$

The rate at which the older values in the series are dampened is controlled by α. If the value of α is close to 1, then the dampening is quick and when α is close 0, the dampening is slow.

The user's profile metrics form the series which varies with job submissions. The exponential predictor operation is very useful when there is sudden change in the bandwidth or memory usage of a user's job. The training period of this predictor is small and hence the predictor quickly adjusts to the changing values.

4 Prediction Quality Metric (PQM)

If most of the metrics recorded in the user's profile are clustered together without any large deviations, then it is fair to say the predictions made will have high accuracy. To calculate the PQM, all the values in the user's profile are compared with the metric value of the first job. The deviation with respect to the metric of the first job is calculated. The PQM is calculated by taking the ratio of number of metrics in the user's profile that have a deviation of less than 20% to the total number of metrics in the user's profile. The PQM value ranges between 0 and 1. If the value is closer to 1, it indicates that most of the user's profile metrics are clustered together and hence the predictors will show good accuracy.

5 Operation of Predictor

The block diagram in Fig 1 shows the steps involved in prediction of performance metrics of a user application. Users jobs submitted to cluster are received by the resource manager TORQUE, scheduled by Moab and run on the allocated service nodes. A monitoring script is run on every node allocated to the job. This script monitors the ethernet statistics and extracts the network parameters of bytes-in and bytes-out by polling the ethernet counters on the machine. The information obtained from the counters is stored along with the system time-stamp as 'network log files'. Hence for each job, a set of 'network log files' are generated on the service nodes on which the job runs.

The resource manager TORQUE, provides a logging option, which logs all events of the job running on the cluster. Some of the information captured by TORQUE logging are user name, start time of job, end time of job, number of processors requested, queue to which the job was submitted and lot of other information. A parser is used to extract the required information from the TORQUE logs and the 'network log files'. The performance metrics namely Per Processor Bandwidth (in bits per second)and Per Processor Memory are calculated for each user's job using the Eq.11 and Eq.12 with the data extracted from the parser. The calculated performance metrics are stored in the SQLite Database.

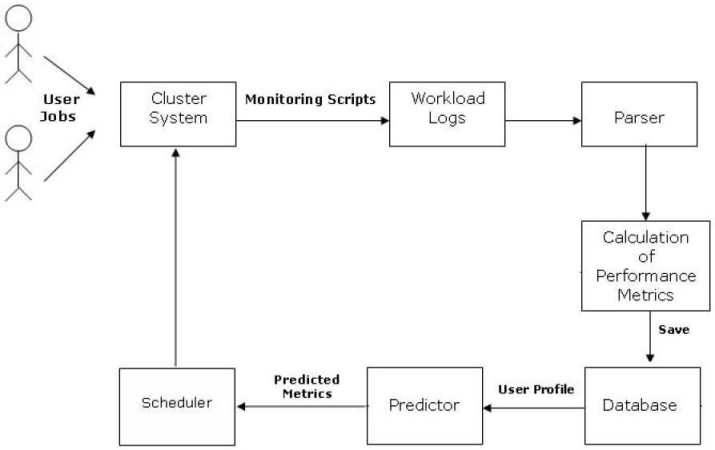

Fig. 1. Predictor Working

$$Bandwidth_{Total} = \frac{Max(\Delta bytes\_in, \Delta bytes\_out) \times 8}{number\ of\ seconds\ in\ \Delta} \tag{10}$$

$$Bandwdith_{perprocessor} = Bandwidth_{Total}/Processor_{count} \tag{11}$$

$$Memory_{perprocessor} = Memory_{Total}/Processor_{count} \tag{12}$$

For each user a profile is created. The user profile has information of each job submitted by the user. For each job the information stored includes user name, job ID, network bandwidth consumed, memory consumed, CPU Utilization, number of processors requested by the job and execution time. When a prediction is to be made, the user's profile is observed and statistical algorithms are applied on the user profile by the predictor to obtain the predicted performance metrics.

6 Experimental Setup

In order to determine the predictability of the performance metrics, the workloads obtained from the Saguaro production cluster at High Performance Computing Laboratory, ASU were analyzed. Overall the test workload consisted of 600 jobs, of which 550 were real jobs from the production cluster Saguaro and about 50 synthetic jobs from the test cluster Agave. In particular we were interested in bandwidth and memory usage of the jobs submitted by the users.

7 Results

In our analysis of the workloads obtained from Saguaro cluster system, the performance metrics namely per processor bandwidth and average per processor memory of jobs submitted by many user's showed uniform values. Considering

the memory first, we found 61% of the users had jobs consuming similar memory usage. In these users, more than 80% of jobs showed a deviation of less than 20% of the memory usage of the first job. This shows that the users submitted same (or similar) jobs on most occasions and hence had similar memory requirements. For the bandwidth, the jobs submitted by users showed more deviations compared to the memory. In the analysis, 40% of the users had jobs consuming similar per processor bandwidth. In these users, more than 80% of the jobs showed a deviation of less than 20% of the bandwidth usage of the first job in the user's profile.

7.1 Behavior of Predictors

The predictor developed predicts five different values for each user based on the user's profile information. Each of these values are obtained by applying a unique statistical algorithms on the user's profile data. Each of these predictive algorithms behaves differently depending on the characteristics of the data set.

The main sources for errors in prediction are:
a) Over estimation by reacting sharply to a spike or peak in the data set.
b) Adjusting to the new value after a sudden increase or decrease in the performance metric.

In the following cases, we consider the performance of the predictors and their accuracy for different properties of data in the user's profiles.

Ideal Case The graph in Fig 2 shows the per processor memory usage of User X's jobs. These jobs show extremely small deviations and consume about 30 MB of memory for each run. The behavior is the similar for a span of more than 30 job submissions. The prediction quality metric (PQM) value for User X's memory profile is 0.89. Since all the user's data is clustered together and have low deviation, the accuracy of all the predictors are above 97%.

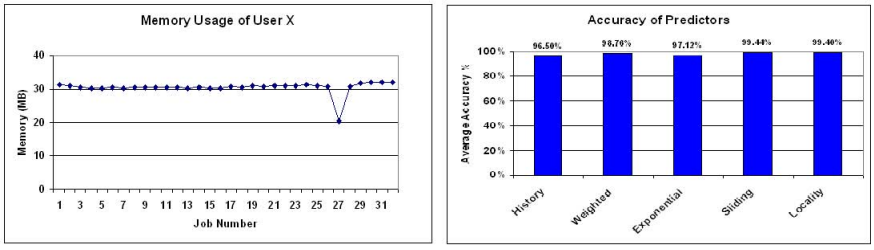

Fig. 2. Most jobs submitted by User X have similar memory usage

User modifies or submits a new application Over a period of time, we expect a user to make changes to his application or submit an entirely new application to the cluster. The new application may have completely different job characteristics. For example: a user may submit an application which performs

lot simulations and once the required data is collected, he may submit a different application that performs data analysis.

The graph in Fig 3 shows the per processor memory usage of jobs submitted by User Y. The memory usage rises suddenly from 200 MB to about 800 MB from Job 2 to Job 3 and remains constant after that. This could be due to the fact that the user changed the application he is submitting to the cluster or modified it largely. The graph in Fig 3 shows the response of the predictors to this change in per processor memory.

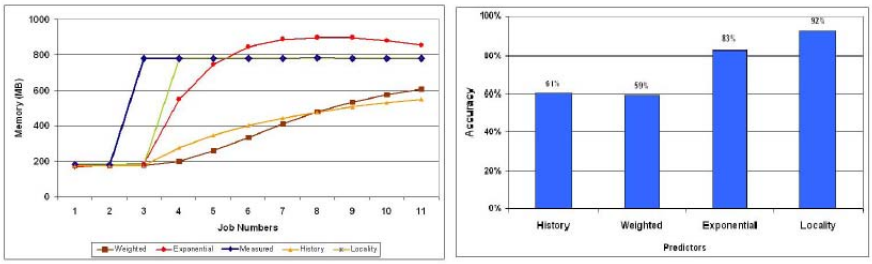

Fig. 3. Predictors reaction to sudden change in per processor memory

The 'History' and 'Weighted' predictors do not predict accurately. The reason being, the history predictor takes the mean of the entire history of jobs, and since change in memory is large and not a steady increase, the history predictor under-estimates the performance metrics and shows low accuracy. The Weighted predictor waits to see if this sudden change in memory is persistent or whether it is just an aberration. Once the change is persistent, it starts to correct itself to predict accurately.

The exponential predictor which calculates the estimated value by assigning larger weights to most recently submitted jobs, reacts to the sudden change and starts to predict accurately within couple of iterations. The exponential predictor over estimates the values of the performance metrics slightly for 2 to 3 iterations as it expects the per processor memory to rise continuously. Since the memory usage after the change is fairly constant, the predictor senses these constant values and corrects itself soon to predict accurately. However, if the value of α in Eq. 6 is reduced, the predictor reacts slower to the trends in data and will not overestimate. The drawback of this being that, it will take more number of iterations to catch up with the sudden change in memory. The Locality predictor, which goes by the value of the most recently submitted job predicts accurately most times except at Job 3. However, if the values betweens successive jobs differ (like between Job 2 and Job 3), the locality predictor forecasts incorrect values.

The graph shows the accuracy of the predictors. It is clear that the history and weighted predictors are not accurate. Hence for users who submit long runs of multiple applications or users who switch between multiple applications, the exponential and locality predictor will provide the most accurate forecasts.

User's Performance metrics distributed over a range of values The graph in Fig 4 shows the per processor memory usage of User Z's jobs. As

observed from the graph, jobs submitted by User Z tend to use varying amount of per processor memory. It is difficult to predict accurately for user's whose profile information contains performance metrics similar to User Z.

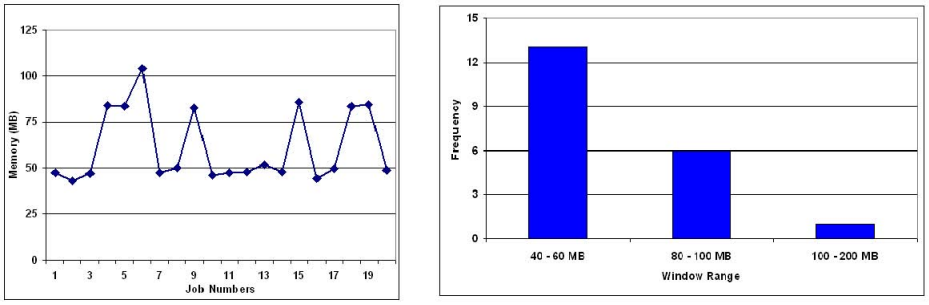

Fig. 4. Per processor memory for User Z's jobs spread over a range of values

The predictor responses for User Z's jobs are show in the graph in Fig 5. Here the per processor memory for last 10 jobs submitted by User Z is predicted by all the five predictors. Before we explore the results of the predictors, a closer look at Fig 4 indicates that almost 70% of the jobs submitted by User Z consume around 50MB of per processor bandwidth.

Now let us consider the results of the predictors as shown in Fig 5. Observing the history predictor results, we see that, though 70% of the jobs consume about 50MB of memory, the values predicted by the history predictor is raised due to the few other jobs that consume high memory (like Job 6 in Fig 4).

The Locality predictor fails on most occasions since since the difference between successive values of memory in User Z execution history is large. The exponential predictor with lower value of α, (α=0.1) for this test predicts around 60MB. Since the value of α is set to a low value, the exponential predictor does not react sharply to the trends in data and hence does not over-estimate the predicted values. Sliding Window Predictor (with 'window size' = 6)predicts reasonably accurate values.

The Weighted predictor, predicts accurately on most occasions with an average prediction value of about 53MB. The weighted predictor does not react much to the seldom occurring high memory utilization jobs. Since 70% of the jobs consume about 50MB, this predictor assigns higher weights to these jobs during the calculation of per processor memory usage.

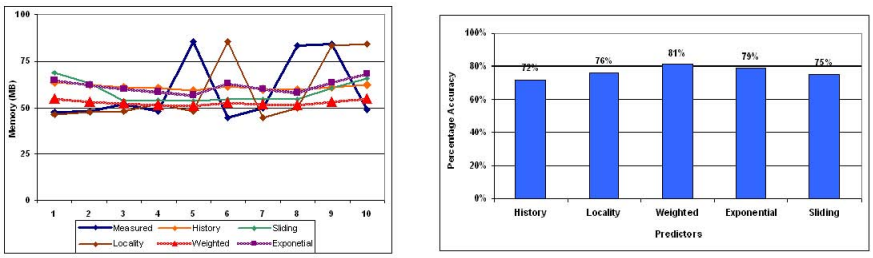

Fig. 5. Predictor Results & Accuracy of Predictors

The accuracy of the predictors are shown in the graph in Fig 5. The weighted predictor has the best accuracy. The exponential predictor and locality predictor also provide good accuracy as the react quickly to the changing data in the user's profile. Hence for jobs whose performance metrics are spread over a range of values and keep fluctuating, the weighted predictor performs well compared to the other predictors.

7.2 Inferences

All the five predictors were used to predict the bandwidth and memory for user jobs. From our analysis of predictor results, we observe that:

a) Predictors using entire execution history for forecasting show reasonable accuracy, for users whose execution history is small or for users who do not make any significant changes to their applications, such that it changes the job characteristics drastically.

b) For users who change or modify their application over a period of time, taking the entire execution history into consideration for prediction is not very accurate. Predictors using limited history or giving higher weights to more recently observed metrics provide better prediction accuracy, as seen in Fig 6.

c) If users job characteristics suddenly change due to modification or change in application, predictors using limited history are able to correct themselves to the sudden change in value. Accuracy may be poor for a couple of iterations till the predictor gets trained to the new value.

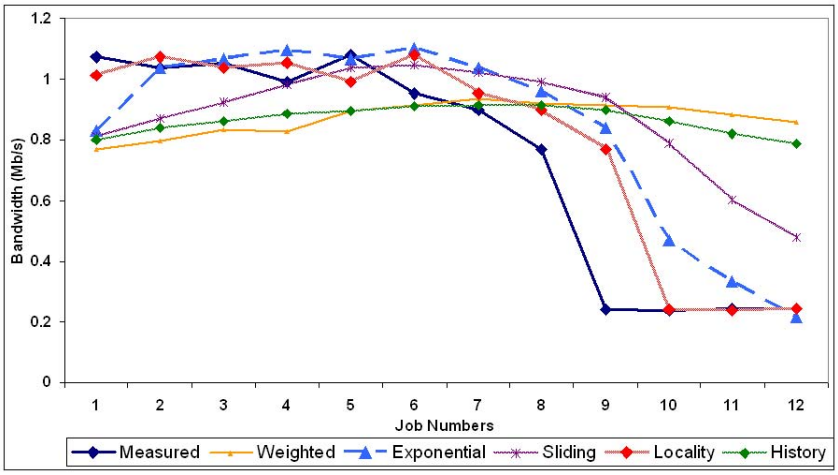

Fig. 6. Predictors Ouputs

Our analysis shows that the History and Weighted predictors do not react sharply to the trends in the measured values and hence their accuracy drops. The Sliding window reacts to the sudden deviations in the metric values, but the behavior is controlled by the size of the 'window' and the values present in the 'window'. Exponential Predictor and the Locality predictor perform well and

have good accuracy of predictions. However, unlike the exponential predictor, the predicted value of the Locality predictor cannot be controlled by any means. When user's performance metrics change frequently, the accuracy of the Locality predictor drops as it reacts to all the changes in the measured values. The Exponential predictor operation can controlled using the value of α in Eq.7. This predictor shows the best average accuracy of about 80% for memory predictions and 72% for bandwidth predictions. For user profiles whose PQM greater than 0.8, the average accuracy was seen as 85% for memory and 80% for bandwidth predictions.

8 Conclusion

This research shows that the resource requirements of a job in a cluster system can be predicted based on the past behavior of the user's submitted jobs. The 'Profile Guided Predictor' tested on real workloads obtained from Saguaro production cluster has shown good accuracy in predicting the bandwidth, memory and CPU Utilization of jobs submitted by users.

The immediate future work of this research is to have the predictor plugged into a meta-scheduler to the test job co-allocation performance in multi-cluster systems. With the predictor providing the required knowledge of job's bandwidth requirements prior to scheduling, the meta-scheduler can make better co-allocation decisions and avoid scheduling job's having high bandwidth requirements across the clusters. This can improve the turn-around time of the jobs, thereby decreasing queue time and increasing the throughput of the system.

References

[1] William M. Jones and Walter B. Ligon and Louis W. Pang and Dan Stanzione, Characterization of Bandwidth-aware Meta-schedulers for Co-allocating Jobs Across Multiple Clusters.
[2] S.Iqbal, R.Gupta and Y.C.Fang, Planning considerations for job scheduling in HPC clusters.
[3] William. M. Jones, The Impact of error in user-provided bandwidth estimates on multi-site parallel job scheduling performance.
[4] A. M. Weil, D. G. Feitelson, Utilization predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling.
[5] H. Li, D. Groep, L. Wolters, Workload characteristics of a multi-cluster supercomputer, in: In Job Scheduling Strategies for Parallel Processing, Springer Verlag, 2004, pp. 176 - 193.
[6] S. Hotovy, Workload evolution on the cornell theory center ibm sp2, in: IPPS 96: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, Springer-Verlag, London, UK, 1996, pp. 27-40.
[7] D. G. Feitelson, B. Nitzberg, Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860, in: Job Scheduling Strategies for Parallel Processing, Springer Berlin / Heidelberg, 1995, pp.337 -360.
[8] R. Wolski, N. T. Spring, J. Hayes, The Network Weather Service: a distributed resource performance forecasting service for metacomputing.
[9] P. A. Dinda, A prediction-based Real-Time Scheduling Advisor, in: IPDPS 02: Proceedings of the 16th International Symposium on Parallel and Distributed Processing, IEEE Computer Society, Washington, DC, USA, 2002, p.10.2.

[10] D. C. Montgomery, E. A. Peck, G. Vining, Introduction to Linear Regression Analysis, 3rd Edition, Wiley, 2003.

[11] H. Li, D. Groep, J. Templon, L. Wolters, Predicting job start times on clusters, in: CCGRID04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Washington, DC, USA, 2004, pp.301308.

[12] W. Smith, V. E. Taylor, I. T. Foster, Using run-time predictions to estimate queue wait times and improve scheduler performance, in: IPPS/SPDP 99/JSSPP 99: Proceedings of the Job Scheduling Strategies for Parallel Processing, Springer-Verlag, London, UK, 1999, pp.202-219.