Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 1 (2010) 2701–2710

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

Jaccard Index based Availability Prediction in Enterprise Grids

Mustafizur Rahman, Md. Rafiul Hassan, Rajkumar Buyya[*]

Department of Computer Science and Software Engineering
The University of Melbourne, Victoria 3010, Australia

Abstract

Enterprise Grid enables sharing and aggregation of a set of computing or storage resources connected by enterprise network, but the availability of the resources in this environment varies widely. Thus accurate prediction of the availability of these resources can significantly improve the performance of executing compute-intensive complex scientific and business applications in enterprise Grid environment by avoiding possible runtime failures. In this paper, we propose a Jaccard Index based prediction approach utilizing lazy learning algorithm that searches for a best match of a sequence pattern in the historical data in order to predict the availability of a particular machine in the system. We compare it against three other well known availability prediction techniques using simulation based study. The experimental results show that our Jaccard Index based prediction approach achieves better prediction accuracy with reduced computational complexity when compared to other similar techniques.

Keywords: Availability, Data Mining, Forecasting, Enterprise Grid, Jaccard Index

1. Introduction

Over the last decade, the Grid [1] has been emerged as a distributed system for sharing and aggregating geographically distributed heterogeneous networked resources in order to provide transparent, dependable, pervasive and consistent computing support to a wide range of applications such as e-Science, e-Business and multimedia application. Grids can be classified in many ways, according to their architecture, presence and services. Considering the presence, Grids are divided into two categories namely, global Grids and enterprise Grids. Global Grids are established over the public Internet and characterized by a global presence, comprising highly heterogeneous resources; whereas enterprise Grids, also known as desktop Grids spread across an enterprise and provide services to the users within that enterprise.

Enterprise Grid [2] enables the selection, sharing and aggregation of a set of computing or storage resources connected by enterprise network, which is under the jurisdiction of a single commercial enterprise or government, educational, or other organization. In an enterprise Grid (refer to Fig. 1), the idle CPU cycles of geographically distributed desktop computers are utilized to run the Grid applications. Therefore, exploiting the unused resources in the enterprise Grid environments across the Internet, can deliver massive compute power to investigate complex and demanding problems in a variety of different scientific fields, such as computational biology, high energy physics, and astronomy. The most well-known examples of enterprise Grid systems are SETI@Home [3], Entropia [4], XtremWeb [5] and Aneka [6].

[*]Corresponding author. Tel.: +61-03-8344-1344.
Email address: raj@csse.unimelb.edu.au (Rajkumar Buyya)

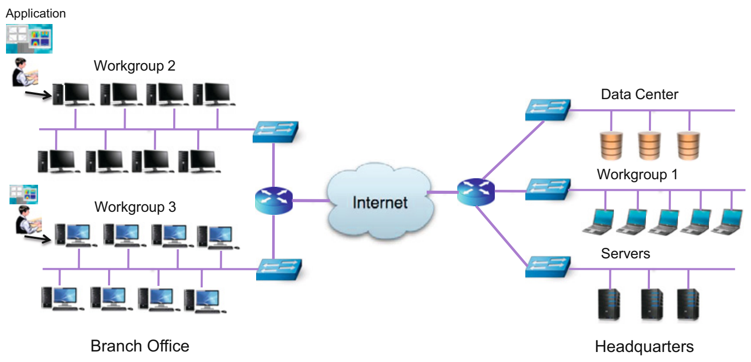

Figure 1: Enterprise Grid Architecture.

1.1. Motivation

Enterprise Grid systems provide some attractive benefits, such as simplicity, access flexibility, ease of deployment and ability to utilize non-dedicated resources. However, one of the significant challenges of such Grid systems is that resources are volatile in nature. The owner of each resource typically exercises ultimate control over the process running on it, its connectivity to the network, and its reboot cycle. As the resources are non-dedicated as well as owned and managed by individuals, the availability of resources in the system changes frequently. This limitation leads to task failure due to the unavailability of the resource; resulting performance degradation of the applications as well as the task schedulers in the system. Therefore, if it is possible to predict the availability of the enterprise wide resources with reasonable accuracy, the application scheduler can plan the mapping of tasks to these resources proactively considering their future availability in order to avoid possible task failures.

Thus, predicting the duration that a machine will run until it restarts (availability duration) is useful to enable effective resource allocation and application scheduling in enterprise Grid systems. In this paper, we examine the problem of predicting machine availability in desktop or enterprise Grid systems and describe a methodology for predicting the availability of enterprise wide resources based on the monitoring data collected from different enterprise computing environments. With the implementation of our proposed methodology, robust prediction accuracy can be achieved even in the highly volatile environment, leading to increased scheduling efficiency.

1.2. Problem definition

The machine availability prediction specifies how long a machine is likely to be available or whether it will be available during a particular period of time. Our goal with this work is to develop a method that can dynamically predict machine availability for the next lookahead period with high prediction accuracy by minimizing incorrect predictions.

Let's consider an enterprise Grid consists of n number of machines. The prediction error, e_i for machine i can be represented as:

$$e_i = \text{Number of incorrect predictions for machine i}$$

If e_i is minimized for every machine, the overall prediction accuracy for the system will be increased consequently. Therefore, the problem of maximizing the availability prediction accuracy of an enterprise Grid system can be formulated as follows:

$$\text{Minimize Error, E} = \sum_{1 \le i \le n} e_i$$

In order to achieve this goal, we propose Jaccard Index based prediction approach utilizing lazy learning algorithm that dynamically searches for a best match of a sequence pattern in the historical or training data in order to predict the availability of a particular machine in the system.

1.3. Contribution

The main contributions of this work include:

- preprocessing of trace data collected from two enterprise Grid computing environments.
- proposing Jaccard Index based novel resource availability prediction technique for enterprise or desktop Grid systems.
- conducting an extensive simulation based study to prove the effectiveness of the proposed approach using real-world traces.
- comparison of the proposed approach against three other well known availability prediction techniques.

1.4. Organization

The remaining of the paper is organized as follows. In the next section, existing availability prediction models and techniques, applied to enterprise Grid systems are described in brief. Section 3 presents the proposed Jaccard Index based prediction methodology along with examples. Section 4 provides the datasets and parameters used in the experiments as well as the experimental setups. Simulation results with respect to comparison of proposed approach against other prediction techniques are discussed in Section 5. Finally, we conclude the paper with the direction for future work in Section 6.

2. Related Work

The main focus of this section is to compare the novelty of the proposed resource availability prediction technique with respect to existing approaches. We classify the related research in two main areas:

2.1. Availability Trace

Several research work in the field of distributed systems describe or analyze traces obtained by logging availability of enterprise Grid resources or individual workstations over time.

A trace containing more than 51,000 machines at Microsoft has been analyzed by Bolosky et al. [7]. This analysis presents the uptime distribution and available machine count over time as well as the correlations among machine uptime and CPU load over the period of a week and lifetime.

Rood et al. [8] analyze the data in a four month Condor [9] resource pool trace at the University of Notre Dame in early 2007. Whereas, Ryu et al. [10] utilize a Condor trace to describe the Fine Grained Cycle Sharing System.

Bhagwan et al. [11] study about 2400 hosts in the Overnet Peer-to-Peer file-sharing system over a period of 7 days. The authors use a prober to periodically probe the hosts in order to determine whether they are available (if responds to the probe) in the system or not at that particular time. Similarly, Chun et al. [12] examine PlanetLab all pairs ping data, consisting of pings sent every 15 minutes between all pairs of 200-400 PlanetLab nodes from January 2004 to June 2005, and report resource availability distribution and mean time to failure.

In this paper, we utilize two empirically gathered availability traces in order to test the performance of our prediction technique. The first trace has been taken from 51,662 PCs in the Microsoft corporate network, and the second captured the behavior of 321 nodes in the PlanetLab distributed testbed. Microsoft data spanned the five weeks period from July 6 to August 9, 1999, whereas PlanetLab data spanned the five weeks period from July 1 to August 4, 2004.

2.2. Prediction model

Ren et al. [13] leverage host CPU utilization and resource contention traces to develop a model for resource availability prediction. The model utilize state transition based prediction and produce a Markov chain using the previous days of a resource's history.

In order to predict successful job execution, Pietrobon and Orlando [14] use a regression based analysis of past/historical record of job executions. Whereas, Nurmi et al. [15] propose an approach to model machine availability based on four statistical distributions: Exponential, Pareto, Weibull and Hyper-exponential as well as evaluate them using goodness-of-fit tests.

On the other hand, we propose Jaccard Index based prediction approach utilizing lazy learning algorithm that searches for a best match of a sequence pattern in the historical or training data in order to predict the availability of a particular machine in the system.

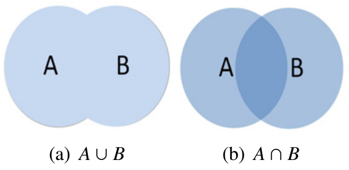

Figure 2: The region of *Union* and *Intersection* between two sets *A* and *B*.

3. Proposed Methodology for Availability Prediction

In this section, we present our methodology for availability prediction. After analyzing enterprise Grid trace data, we find that the availability trend in such environment is random in nature and changes continuously; making it very difficult to identify the underlying distribution of the system. Lazy learning techniques do not require to build a prior model and do not need to identify the underlying data distribution. Further, lazy learning algorithm is adaptive in nature and it can dynamically update the training or historical data to generate a prediction. Thus, lazy learning technique is suitable for predicting the availability of computing machines in enterprise Grid environment and we propose Jaccard Index based availability prediction approach utilizing lazy learning algorithm. In the next subsections, first we give a brief description of the concept behind Jaccard Index. Then we discuss about how Jaccard Index has been utilized in our technique to perform availability prediction using examples and algorithms.

3.1. Jaccard Index

The Jaccard Index (JI), also known as the Jaccard Similarity Coefficient [16] [17], is a measurement that is used for identifying the degree of similarity and diversity of two data windows. Let us consider two data windows, X_i and X_j. The coefficient measures the degree of overlap between two windows by computing the ratio of the number of shared attributes between X_i and X_j. For simplicity, let us consider two sets A and B in place of data windows. As depicted in Fig. 2, the region of *Intersection* ($A \cap B$) and *Union* ($A \cup B$) between these two sets can be measured according to set theory. Thus, the Jaccard index is calculated as follows:

$$Jaccard(A, B) = \frac{A \cap B}{A \cup B} \tag{1}$$

To identify the similarity between two data windows, X_i and X_j the *Intersection* between these windows is obtained by calculating the cardinality of same value attributes in the windows as depicted in Eq. 2.

$$\vec{X_i} \cap \vec{X_j} = \sum_{k=1}^{n} X_{k_i} \cap X_{k_j}, \qquad X_{k_i} \in \vec{X_i} \quad \text{and} \quad X_{k_j} \in \vec{X_j} \tag{2}$$

where,

$$X_{k_i} \cap X_{k_j} = \begin{cases} 1, & \text{if } X_{k_i} = X_{k_j}; \\ 0, & \text{otherwise.} \end{cases}$$

In standard set theory, the *Union* of the two sets is the set of all distinct elements in the sets. While calculating the Jaccard similarity measurement, the *Union* between two data windows $\vec{X_i}$ and $\vec{X_j}$ is obtained by using Eq. 3.

$$\vec{X_i} \cup \vec{X_j} = \sum_{k=1}^{n} X_{k_i} \cup X_{k_j}, \qquad X_{k_i} \in \vec{X_i}\ ; X_{k_j} \in \vec{X_j} \text{ and } X_{k_i} \cup X_{k_j} = 1 \tag{3}$$

$$= \text{Dimension of the data window} \tag{4}$$

Thus, the Jaccard similarity measurement or Jaccard Index, $Jaccard(\vec{X_i}, \vec{X_j})$ between the two windows $\vec{X_i}$ and $\vec{X_j}$ is:

$$Jaccard(\vec{X_i}, \vec{X_j}) = \frac{\text{cardinality of } Intersection}{\text{cardinality of } Union} \tag{5}$$

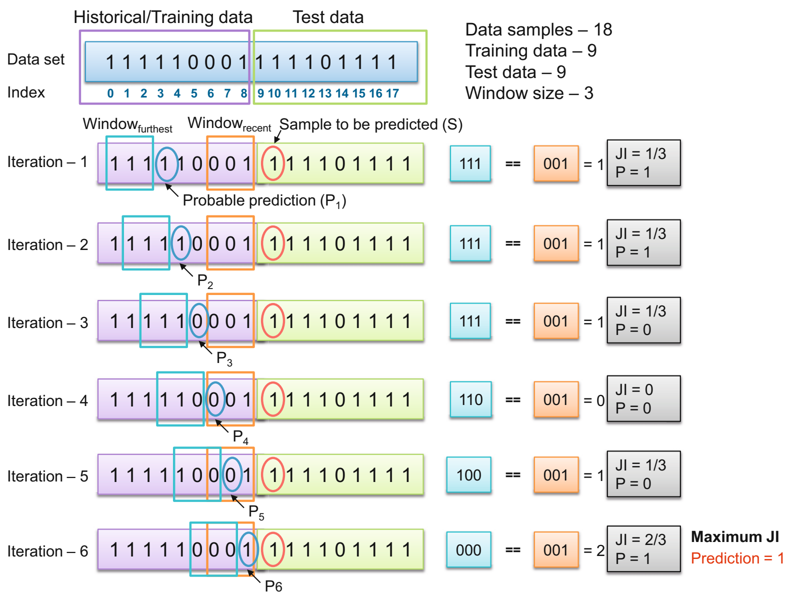

Figure 3: Example of Jaccard Index based availability prediction for enterprise Grid resources. Here, operation ′ ==′ calculates the number of dissimilar data elements in the operators.

3.2. Jaccard Index based Prediction

3.2.1. Jaccard Index Computation

The machine availability trace data that is used for prediction consists of a sequence of zeros and ones, where zero represents unavailability and one represents availability of a particular machine. For such a binary data the Jaccard Index (JI) is calculated as follow:

- Given a sequence of training or historical data, we compute most recent and furthest data windows ($Window_{recent}$, $Window_{furthest}$).
- Then we calculate JI for $Window_{furthest}$ with respect to $Window_{recent}$. At this step, the window that has JI equal to 1 signifies the exact match to $Window_{recent}$.
- Once JI for the first $Window_{furthest}$ is calculated, we slide the window to right by one element to get new $Window_{furthest}$ and calculate JI for the new window. This process is continued until the window reaches the second last elements of training data.
- Then the window for which the highest JI has been calculated is selected and value of the data element just after this window is considered as the predicted value for the current availability point.
- Once the next data point or availability observation that has been predicted is available, $Window_{recent}$ is shifted right by one element and prediction for the next lookahead period is determined in the same way as mentioned above.

These steps are further explained in Fig. 3. The example illustrated in this figure considers a data sample of 18 availability observations from enterprise Grid, where 9 are used as historical or training data and rest are used as test data. Initially, the sequence of training data is divided using a window size of 3 (i.e. each data window has 3 consecutive data elements). The value of each data element can be either 0 or 1. As shown in the figure pointed by *iteration 1*, the most recent window ($Window_{recent}$) in the training data consists of the 3 elements, $\langle 0, 0, 1\rangle$ and the farthest data window ($Window_{furthest}$) consists of the 3 elements, $\langle 1, 1, 1\rangle$. Since only one element between these two windows are similar, according to Eq. 5, the JI for these two data windows is:

$$Jaccard(\vec{X_i}, \vec{X_j}) = \frac{1}{3}$$

Now, the availability prediction (P) for this window is the value of the data element right after the window, which is 1. So if JI of this window is the highest in comparison to the subsequent prediction windows, availability prediction for the first test data element (S) will be P_1 that is 1. Thus, it will be a correct prediction.

In the next iteration, $Window_{furthest}$ is shifted towards right by one element and JI for this window with respect to $Window_{recent}$ is P_2, which is also 1. In Fig. 3, the maximum JI among all the windows is $\frac{2}{3}$ for the data window $\langle 0, 0, 0 \rangle$. Therefore, the value of next element after this window from the training data, P_6 is considered as the predicted value for S.

3.2.2. Incorporating Voting

The window size for the example shown in Fig. 3 is fixed to 3. However, if the window size is varied, the outcome for the prediction value can also be varied. Thus in order to improve the prediction accuracy, for a single prediction, we vary the window size and get a prediction value for the corresponding window size. In that case, a majority voting among the predicted values is applied to generate a single prediction value.

Let us consider 5 windows of different sizes, X_i, X_j, X_k, X_l and X_p that have the same JI, j. The prediction value, P of X_i is 1, X_j is 0, X_k is 1, X_l is 1 and X_p is 0. Since, majority of the prediction values are 1 (3 out of 5), we consider the prediction value in regards to the most recent window X_{recent} is 1. In Fig. 3, window size is not changed and hence no majority voting is applied to generate the prediction. Once the next data point or availability observation that has been predicted is available, the most recent window is slided towards right by one element to generate prediction for the next lookahead period. Hence, the method is an online adaptive algorithm that adapts with the newly arrived data.

3.2.3. Computational Complexity

Let us consider, the training data consists of N elements and window size is fixed to W. Then according to Algorithm 1, in order to predict availability value for a single data point, there will be $N - W$ iterations and JI is calculated in each iteration. We compare every element of $Window_{recent}$ and $Window_{furthest}$ for calculating JI. Thus, the computational complexity of the proposed Jaccard Index based prediction technique is $O(NW - W^2)$ for predicting a single availability observation.

4. Datasets and Experiment Design

4.1. Dataset

In order to determine the real-world applicability of our availability predictor, we test it on two empirically gathered availability traces. The availability behaviors of these two traces are explained in [18]. The first trace has been taken from 51,662 PCs in the Microsoft corporate network [7] and the second captured the behavior of 321 nodes in the PlanetLab distributed testbed [19]. Each machine in the Microsoft trace was pinged hourly. In the PlanetLab traces [20], machines were pinged every 15 minutes, but we sampled every fourth measurement to provide a fair comparison with the Microsoft data. The lifetimes of PlanetLab nodes were long enough that such sampling did not distort the underlying availability patterns. Microsoft data spanned the five weeks period from July 6 to August 9, 1999, whereas PlanetLab data spanned the five weeks period from July 1 to August 4, 2004.

For evaluating the predictor on Microsoft and PlanetLab data, initially we use first two weeks of the traces as historical data, which can also be considered as the train data. We evaluate the accuracy of our predictor using the remaining three weeks of the availability data. At a sample rate of once an hour, this results in 336 training samples and 504 testing or evaluation samples. During each hour in the evaluation period, the predictor makes forecast for the next lookahead interval or sample. After each prediction, the historical/training data is updated by adding this sample and deleting the least recent sample, while keeping the length fixed. Then the forecast for the next lookahead interval is performed using the updated training data. This makes the process of forecasting dynamic and improves accuracy, while minimizing runtime complexity.

Algorithm 1: Predict

```
Input: $\mathcal{Y}, \omega$; The set of examples: $\mathcal{Y}$; Window size: $\omega$
Output: $y\prime$: The prediction for the next event
$\mathcal{D} \leftarrow$ Process($\mathcal{Y}, \omega$);
$y^* \leftarrow$ The last data vector in $\mathcal{D}$;
$y^{match} \leftarrow$ FindJaccard($\mathcal{D}, y^*$);
$y\prime \leftarrow y^{match+1}(1)$;
Return $y\prime$;

function Process ($\mathcal{Y}, \omega$);
N $\leftarrow$ length($\mathcal{Y}$)-$\omega$ + 1;
for $i = 1 : N$ do
    $\mathcal{D}(i) \leftarrow \mathcal{Y}(i : i + \omega - 1)$;
Return $\mathcal{D}$;

function FindJaccard($\mathcal{D}, y$);
N $\leftarrow$ length($\mathcal{D}$)-1;
for $i = 1 : N$ do
    $Index(i) \leftarrow$ JaccardIndex($\mathcal{D}(i)$ , $y$);
Return $max(Index)$;

function JaccardIndex($x, y$);
numerator $\leftarrow$ 0;
denometor $\leftarrow$ 0;
for $j = 1 : \omega$ do
    if $x(j) = y(j)$ then
        numerator $\leftarrow$ numerator + 1;
    denometor $\leftarrow$ denometor + 1;
$Index \leftarrow \frac{numerator}{denometor}$;
Return $Index$;
```

Algorithm 2: PredictwithVoting

```
Input: $\mathcal{Y}, \Omega$; The set of examples: $\mathcal{Y}$; Window-size list: $\Omega$ ($\omega \in \Omega$)
Output: $y_{pred}$: The prediction for the next event
y $\leftarrow \phi$;
N $\leftarrow$ length($\Omega$);
for $i = 1 : N$ do
    y(i) $\leftarrow$ Predict($\mathcal{Y}, \omega(i)$);
if $count(y = 1) > count(y = 0)$ then
    $y_{pred} = 1$;
else
    $y_{pred} = 0$;
Return $y_{pred}$;
```

4.2. Performance Metrics

As a measurement of the performance of our proposed Jaccard Index based availability prediction technique, we evaluate the following performance metrics:

ACC_{avg}**:** We measure it as the average prediction accuracy of all the machines. Thus, if an enterprise Grid consists of n number of machines and prediction accuracy of machine i is $Accuracy_i$, then ACC_{avg} can be expressed as,

$$ACC_{avg} = \frac{\sum_{1\leq i\leq n} Accuracy_i}{n}$$

where,

$$Accuracy_i = \frac{\text{Number of correct predictions for machine i}}{\text{Number of predictions for machine i}} \times 100\%$$

ACC_{95}**:** It is measured as the percentage of machines predicted with greater than 95% accuracy. If there are n number of machines in an enterprise Grid and $\sum_{1\leq i\leq n} Accuracy_i(95)$ is the number of machines predicted with greater than 95% accuracy, then ACC_{95} can be expressed as,

$$ACC_{95} = \frac{\sum_{1\leq i\leq n} Accuracy_i(95)}{n} \times 100\%$$

where,

$$Accuracy_i(95) = \begin{cases} 1, & \text{if prediction accuracy for machine i is greater than 95\%;} \\ 0, & \text{otherwise.} \end{cases}$$

4.3. Experiments

We conduct several experiments to determine the accuracy of our proposed Jaccard Index based availability prediction technique. In the first experiment, we test the performance of proposed technique for PlanetLab trace data using the above mentioned performance metrics, ACC_{avg} and ACC_{95}. In contrast, Microsoft trace data is utilized in the second experiment with regard to performance measurements.

Furthermore, we perform the same set of experiments for three other prediction techniques namely, k-nearest neighbors algorithm, Naive Bayes algorithm and Hybrid predictor. K-Nearest Neighbors algorithm (k-NN) [21] is a type of instance-based learning or lazy learning technique, where the function is only approximated locally and all computation is deferred until classification. The Naive Bayes algorithm [21] is also a lazy learning technique that is based on conditional probabilities. Naive Bayes uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data.On the other hand, Hybrid predictor proposed by Mickens et.al. [18] focuses on dynamically selecting the best predictor for a given availability or uptime pattern and lookahead period by combining five sub-predictors, namely RightNow, SatCount, History, TwHistory and Linear predictor. The final output of the Hybrid predictor is selected via tournaments, where tournament counters determine the best predictor to use for a particular lookahead period. This helps to track fine-grained, per-node uptime state to estimate future availability, leveraging the most accurate estimation mechanism for each situation.

5. Results and Discussion

Table 1 and Table 3 present the average prediction accuracy for lazy learning based prediction approaches (Jaccard, k-NN, Naive Bayes) on PlanetLab and Microsoft data respectively. The performances of these lazy techniques against Hybrid technique with respect to the above mentioned performance metrics are summarized in Table 2 and Table 4.

From Table 1, it is evident that overall prediction accuracy of all the lazy techniques are greater than 99% for PlanetLab dataset. The reason for such a high prediction accuracy is due to the adaptive nature and non-dependence on data distribution of lazy algorithms. Lazy algorithms are adaptive since it can dynamically update the training data to generate a prediction. Further, lazy algorithms find out the similar patterns from the training dataset and generate a prediction using frequency count among the most similar patterns. Hence, lazy algorithms are not dependent on the data distribution.

As shown in Table 1, the performance of Naive Bayes is slightly worse than that of k-NN and JI based predictors. This is because, Naive Bayes generates prediction considering each of the attribute distributions in the training dataset

Table 1: Comparison of overall prediction accuracy for different window sizes (Planet Lab data)

Window size	Accuracy (%)		
	Jaccard	k-NN	Naive Bayes
3	**99.7379**	**99.7375**	**99.7357**
4	99.7370	99.7357	99.6938
5	99.7370	99.7229	99.6700
6	99.7370	99.7145	99.6215
7	99.7370	99.7113	99.5822
8	99.7370	99.7108	99.5477
9	99.7370	99.7105	99.5122

Table 2: Comparison of best results for each prediction technique (Planet Lab data)

Accuracy (%)	Jaccard Voting	Jaccard	k-NN	Naive Bayes	Hybrid
ACC_{avg}	99.7445	99.7379	99.7375	99.7357	–
ACC_{95}	100.00	100.00	100.00	100.00	95.60

Table 3: Comparison of overall prediction accuracy for different window sizes (Microsoft data)

Window size	Accuracy (%)		
	Jaccard	k-NN	Naive Bayes
3	**96.7759**	**96.2526**	**96.1754**
4	96.7759	96.2526	95.8744
5	96.7758	96.2248	95.8744
6	96.7757	96.2248	95.6615
7	96.7757	96.2035	95.6615
8	96.7755	96.1928	95.2871
9	96.7755	96.1928	95.2871

Table 4: Comparison of best results for each prediction technique (Microsoft data)

Accuracy (%)	Jaccard Voting	Jaccard	k-NN	Naive Bayes	Hybrid
ACC_{avg}	96.8795	96.7759	96.2526	96.1754	–
ACC_{95}	82.2813	81.5941	84.2000	83.87	87.00

Table 5: Comparison of computational complexity for each prediction technique

Jaccard	k-NN	Naive Bayes	Hybrid
$O(NW - W^2)$	$O(N \log N + NW)$	$O(N \log N + NW + W)$	$O(M(N^3 + N2^W))$

and can susceptible to overfitting problem. Interestingly, JI based algorithm performs as good as the other algorithms with a reduced computational complexity (refer to Table 5). For all the lazy algorithms, we notice that the average prediction accuracy is decreased with the increase in window size.

The average prediction accuracy for the JI based prediction method on Microsoft data is also slightly better than k-NN and Naive Bayes (see Table 3). The decreasing performance trend is also evident for lazy algorithms on this dataset with varying window size. It is worth mentioning that JI based prediction outperforms k-NN and Naive Bayes while comparing the total number of machines (in percentage) for which the respective method has the prediction accuracy greater than 95%.

For a further performance evaluation, we compare our prediction accuracy with that of reported by Mickens et al. [18] using Hybrid method. As shown in Table 2, for lazy algorithms, 100% machines are predicted with more than 95% prediction accuracy in comparison to 95.6% for Hybrid predictor on PlanetLab data. Since, average prediction accuracy (ACC_{avg}) is not reported in [18], we compare lazy algorithms against Hybrid predictor with respect to (ACC_{95}).

Table 5 shows the computational complexity of all the availability prediction techniques considered in this paper. It can be seen that our proposed JI based technique has the least computational complexity in comparison to other lazy algorithms as well as Hybrid predictor. Hybrid predictor predicts an availability observation using a tournament among five sub-predictors and yields relatively higher computational complexity. The individual computational complexity of the five predictors are listed in Table 6.

Table 6: Computational complexity of sub-predictors used in Hybrid predictor

RightNow	SatCount	History	TwiddledHistory	Linear
$O(1)$	$O(1)$	$O(NW)$	$O(N(2^W + 1))$	$O(N^3)$

6. Conclusion and Future Work

This paper focuses on presenting an availability prediction technique for computing machines in enterprise Grid environment. The availability trend in such environment is random in nature and changes continuously. Therefore, we propose Jaccard Index based availability prediction approach utilizing lazy learning algorithm. We compare our technique against two state-of-the-art lazy learning algorithms (k-NN, Naive Bayes) and a hybrid predictor using simulation based study. The experimental results show that proposed Jaccard Index based prediction technique achieves accuracy of 99.74% for PlanetLab data and 96.87% for Microsoft data with reduced computational complexity when compared to other prediction techniques..

In future, we endeavor to use this prediction technique to devise an availability-aware Grid application scheduling approach. This will eventually improve the performance of executing complex scientific and business applications such as workflows in enterprise Grid environment.

7. References

[1] M. L. Bote-lorenzo, Y. A. Dimitriadis, E. Gmez-snchez, Grid characteristics and uses: A grid definition, in: Proceedings of 1st European Across Grids Conference, Spain, 2003.

[2] K. Nadiminti, R. Buyya, Enterprise grid computing: State-of-the-art, Technical Report, GRIDS-TR-2005-16, Grid Computing and Distributed Systems Laboratory, The University of Melbourne.

[3] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, D. Werthimer, Seti@home: An experiment in public-resource computing, Communications of the ACM 45 (11) (2002) 56–61.

[4] A. Chien, B. Calder, S. Elbert, K. Bhatia, Entropia: Architecture and performance of an enterprise desktop grid system, Journal of Parallel and Distributed Computing 63 (2003) 597–610.

[5] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. Neri, O. Lodygensky, Computing on large scale distributed systems: Xtremweb architecture, programming models, security, tests and convergence with grid, Future Generation Computer Systems 21 (3) (2005) 417–437.

[6] X. Chu, K. Nadiminti, C. Jin, S. Venugopal, R. Buyya, Aneka: Next-generation enterprise grid platform for e-science and e-business applications, in: Proceedings of 3rd IEEE International Conference on e-Science and Grid Computing (e-Science'07.

[7] W. Bolosky, J. Douceur, D. Ely, M. Theimer, Feasibility of serverless distributed file system deployed on an existing set of pcs, in: Proceedings of ACM SIGMETRICS, Santa Clara, USA, June 2000.

[8] B. Rood, M. J. Lewis, Multi-state grid resource availability characterization, in: Proceedings of 8th IEEE/ACM International Conference on Grid Computing (GRID07), Austin, USA, September, 2007.

[9] T. Tannenbaum, D. Wright, K. Miller, M. Livny, Condor - a distributed job scheduler, in: T. Sterling (Ed.), Beowulf Cluster Computing with Linux, The MIT Press, USA, 2002.

[10] K. Ryu, J. Hollingsworth, Unobtrusiveness and efficiency in idle cycle stealing for pc grids, in: Proceedings of 18th International Parallel and Distributed Processing Symposium (IPDPS04), Santa Fe, USA, April, 2004.

[11] R. Bhagwan, S. Savage, G. Voelker, Understanding availability, in: Proceedings of 2nd International Workshop on Peer-to-Peer Systems, Berkeley, USA, February, 2003.

[12] B. Chun, A. Vahdat, Workload and failure characterization on a large-scale federated testbed, Technical Report, IRB-TR-03-040, Intel Research Berkeley.

[13] X. Ren, R. Eigenmann, Empirical studies on the behavior of resource availability in fine-grained cycle sharing systems, in: Proceedings of 35th International Conference on Parallel Processing, Columbus, USA, August, 2006.

[14] V. Pietrobon, S. Orlando, Performance fault prediction models, Technical Report, CS-2004-3, Department of Computer Science, University of Venice.

[15] D. Nurmi, J. Brevik, R. Wolski, Modeling machine availability in enterprise and wide-are distributed computing environments, Technical Report, CS2003-28, UCSB Computer Science.

[16] A. H. Cheetham, J. E. Hazel, Binary (presence-absence) similarity coefficients, Journal of Paleontology 43 (5) (1969) 1130–1136.

[17] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, Communications of the ACM 51 (1) (2008) 117–122.

[18] J. Mickens, B. Noble, Exploiting availability prediction in distributed systems, in: Proceedings of 3rd Symposium on Networked Systems Design and Implementation (NSDI06), USA, May 2006.

[19] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, M. Wawrzoniak, Operating system support for planetary-scale services, in: Proceedings of 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI'04), San Francisco, USA, March, 2004.

[20] J. Stribling, All-pairs planetlab ping data, `http://www.pdos.lcs.mit.edu/strib/pl app/`.

[21] D. W. Aha, Special ai review issue on lazy learning, Artificial Intelligence Review 11.