CPU load prediction based on a multidimensional spatial voting model

Chen Yu
Shanghai Jiaotong University
P.R.China
yucenyucen@126.com

Jian Cao
Shanghai Jiaotong University
P.R.China
cao-jian@sjtu.edu.cn

Pinglei Guo
Shanghai Jiaotong University
P.R.China
pvguo@sjtu.edu.cn

***Abstract*—Resource performance prediction has become more and more important in cloud environment as CPU load prediction is key for system maintenance and application schedule.This paper presents a multidimensional spatial voting prediction model to predict real-time CPU load accurately. We improved the real-time CPU load prediction accuracy by gray prediction model under the one-dimension prediction; we also applied voting mechanism to find a more appropriate classifier prediction model for predicting the CPU load in real time. Our experiments showed that multidimensional spatial voting prediction model led to better predictions than classic models. Our model is not problem-specific, and can be applied to problems in the fields of other predictions.**

Keywords—multidimensional spatial voting, gray prediction model, CPU load prediction

I. INTRODUCTION

With the development of computer science, Computer has become an important and indispensable resource just like water, electricity and gas. During last decade, distributed system has prevailed around the world. Distributed system is a set of independent computers which are connected together and work collaboratively to solve one problem. Resources such as CPU load, memory load and remaining space of each disks vary from time to time. The Resource is no longer static but dynamic but our scheduler uses the ideal static algorithm to manage tasks according to the information collected may be several minutes and deems all the resource are still same. In this circumstance, prediction can be a good solution.

Nowadays, common prediction methods include: multi factor line regression method, trend forecast, seasonal moving average, exponential smoothing and so on. Although there are a lot of prediction methods, not every method can achieve good results in the field of CPU load forecasting. In the field of CPU load forecasting, it is difficult to choose a method as CPU load of different computer exhibits completely different properties. In this situation, Bates JM and Granger CWJ[1] created the concept of integrated forecasting in 1969. They[2] combined different kinds of prediction algorithm to improve universality of the algorithm, increasing accuracy of prediction by choosing the most appreciate method in specific situation. However, it's hard to judge the weight of every single model in integrated forecasting models. What's more, the accuracy can even be lower than classic prediction models in systems which are random and nonlinear. From another perspective[3], integrated forecasting are generally one-dimension, which uses only one kind of historical data. But in the field of CPU load predicting, CPU load is not relevant to its historical data only[4] , but also to many other factors such as memory usage, disk IO, network IO, etc.

II. RELATED WORK

In the past twenty years, predictions on CPU load has been widespread concerned. Researchers have proposed and implemented a number of prediction models. However, due to the complexity and dynamic nature of the environment, robustness, stability and accuracy of the algorithm should be improved. According to resources involved in predicting, these models can be classified into unidimensional prediction and multidimensional prediction.

A. Unidimensional prediction

The most famous unidimensional prediction is exponential smoothing(ES)[5]. It was raised by Robert G..Brown), Brown, believes the trend of the time series with the stability or regularity, the time sequence can be reasonably postponed homeopathy; he thinks the recent past trend will continue in the future in a way.

Peter A. Dinda et al.[6][14] used linear CPU load prediction models to evaluate 1-30 seconds CPU load based on five-second CPU average load data. These models include auto-regressive model (AR), moving average model (MA), auto-regressive moving average model (ARMA), autoregressive integrated moving average model (ARIMA) and a long memory time series based on autoregressive fractionally integrated moving average model (ARFIMA) and so on. In these models, AR model is most widely used in CPU load prediction.

AR model can be applies to dataset within a certain period of time, and is extremely stable. Peter A. Dinda's experiments showed that in some cases, linear model can be used to predict CPU loads and AR proformed better. AR model was stable with lower computational complexity, while other models spent much time. Linear model prolonged the prediction time to five minutes, but it was still insufficient.

B. Multidimensional prediction

Resource itself is not the only factor to change its status, impact of other resources also play an important role. So we can do some deduction by analyzing changes of other resouces. Status of these resources can be used as a variable of the prediction algorithm to calculate the prediction value to achieve the accuracy of the predictions.

M.Swany[21][22] described different kinds of multidimensional prediction technologies which involved the interaction between the resource to predict and other factors. The algorithm designed a series of polynomial fitting to indicate the relationship and improved the accuracy of prediction. Although these techniques did not involve CPU load prediction, their research gave us some inspiration.

Based on previous studies, J.Liang[24] developed a new model called multi-resource prediction model(MModel). The model invited not only correlation of historical data of resource, but also interaction of different resources (such as CPU load and free memory space). MModel is adaptive, as it can collect real-time resource status changes over time. Experiment results showed that the accuracy of MModel, by which prediction error was reduced by 6% - 90%, was much higher than AR model.

III. GRAY PREDICTION MODEL

When historical data is periodic and regular, classic model meets our requirements very well. Even when historical data is not obviously periodic, Exponential Smoothing model[6] can do a good job. But problems of historical data such as limited sample size, little regularity and noise have been revealed in real-time computer load forecasting. Therefore, it is necessary to find the key part of historical data and its regularity to provide an important basis for the next stage. That's the reason why gray prediction model[7][8] is under our consideration.

The experiment data showed below comes from some company servers[9]. The data includes real CPU load and three related variables(memory usage, disk IO, net IO)[10]. Sampling time is once per minute. The CPU load data reflected the real situation objectively as it was obvious regular or with little regularity, sometimes unusual. We applied gray prediction model to predict the three relevant variables. As raw data of these variables came from dynamic computer server system is not regular, firstly we used buffer operator to correct them so that the value became gentle and fit for gray prediction model. In this paper, XD_m is defined as the data being corrected by buffer operator, in which m presents the times that buffer operators worked, and m = 1,2, ..., n. It is derived as follows:

The raw data is:

$$X^{(0)} = (x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)})$$

Among them

$$x(k)d^2 = \frac{1}{n-k+1}[x(k)d + x(k+1)d + \cdots + x(n)d], k = 1,2,\cdots n$$

We used 1-AGO to deal with the data series optimized by buffer operator to get $X^{(1)}$

$$X'^{(1)} = (x_1'^{(1)}, x_2'^{(1)}, \cdots, x_n'^{(1)})$$

Test the smoothness of the new data series:

$$\rho_k = \frac{x_k'^{(0)}}{x_{k-1}'^{(1)}} = \frac{x_k'^{(0)}}{\sum_{i=1}^{k-1} x_i'^{(0)}}$$

Coefficient *a* depends on smoothness value ρ .

We differentiated $X^{(1)}$:

$$\frac{dx_1'^{(1)}}{dt} + ax_1'^{(1)} = b$$

Two parameters of gray prediction model satisfied Least Squares :

$$(a,b)^T = (\mathbf{B}^T, \mathbf{B}^{-1}\mathbf{B}^T\mathbf{Y})$$

Among them

$$B = \begin{bmatrix} -z_2^{(1)}, 1 \\ -z_3^{(1)}, 1 \\ \vdots \\ -z_n^{(1)}, 1 \end{bmatrix}, Y = \begin{bmatrix} x_2'^{(0)} \\ x_3'^{(0)} \\ \vdots \\ x_n'^{(0)} \end{bmatrix}$$

The final one-dimension prediction model is:

$$\hat{x}_{i+1}'^{(0)} = \hat{x}_{i+1}'^{(1)} - \hat{x}_i'^{(1)} = (1-e)(x_1'^{(0)} - \frac{b}{a})e^{-ak}$$

IV. MULTIDIMENSIONAL SPATIAL VOTING PREDICTION MODEL

A. the input and output of the classifiers

To a classifier, input is an eigenvector[11] and output is the current classification of data. CPU load data is continuous in the time, but computer is a discrete system. So frequent sampling leads to discrete data. Theoretically, the range of CPU load data is 0 to infinity, but it is presented by limited data points.

Based on these assumptions, first we chose the continuous CPU load data of length L before the prediction point as the input eigenvector[12] and the output label as the prediction value.It was possible that the output label is prediction value as the amount of CPU load data was limited.

As previously mentioned, CPU load data is related to some other indicator. We also added the related indicators to eigenvectors, which included prediction of memory usage, hard disk IO,network IO.

B. voting prediction model

In general, Ensemble Learning[13] is comprised of two steps. The first step is model selection which chooses a better set of algorithms from selected candidate based on a subset of historical data.The second step is model ensemble which calculates the final prediction results based on the weighted candidate sets.

The Ensemble Learning algorithm based on voting derived from human affairs handling. Agree to vote or not represents

the binary 1 and 0, then ensemble learning overlay and judgment m akes the weight between 0-1, make a reasonable decision at last. The CPU load data can be any positive number (theoretically). Another point to consider is that the error of classification must be within 100 percent. As CPU load data can be any value, the MER is can be more than 1.

The first step of the algorithm is using subset of the historical data to calculate error which is the basis of voting decision.Only in the case of error being objective, the Ensemble Learning takes advantages only when error is objective, or it will enlarge the error. Our model invited cross-validation to ensure that the error is not too large.

Voting is based on the error. The number of votes obtained by each algorithm is not integer but decimal, and the sum votes is 1. The calculation method is as follows:

- set the voting number of the each classifier algorithm to zero if its error is higher than the average
- the remaining classifier algorithm will take the normalized reciprocal of its error as its voting weight.

The advantage of voting method is that it don't vote for minimum error algorithm arbitrarily. When errors of the algorithm are close, it is not robust and adaptable to choose only one, considering in the first step we only made a finite number of cross-validation, which may not involve all of the real data. But if we select all the algorithms, it is bound to make the overall error larger. So the first step is removing the algorithm with a greater error than average. It can effectively avoid the impact by algorithm with poor adaptability of the current data. It is a compromise between accuracy, robustness and adaptability. We selected error reciprocal[14] to enlarge the gap between prediction algorithm by weighing the ones with less error more to improve performance.

For the last step, we used dynamic weight superposition to calculate the final prediction value.

C. classifier prediction model based on voting process

Step1: Construction of training data

Input: $\{t_1 , t_2 , t_3 , \dots , t_n, \ t_{memory},t_{disk},t_{net}\}$

Output :$\{\{input, output \}| input=\{t_i , t_i+1 , t_i+2 , \dots , t_{i+L}\}, output= t_{L+i+1}, i = [0 , L] ,i\in z\}$

T_n: CPU load on Time T.

$t_{memory},t_{disk},t_{net}$: usage of the memory, disk, net.

Step2: Train the classifier we used.

$C_{trained}$ = f(C, input, output);

$C_{trained}$: the classifier that have been trained.

The classifier of classifier prediction model based on voting includes KNN, J48, SVM, BayesNet.

Step3: Predict by voting. Pseudo-code is as follows:

1. If not running for the first time

2. For each predictor in Candidate Set

3. Calculate the error rate last time

4. End for

5. Update the scores for each predictor

6. If score of current representative predictor < FLOOR LIMIT

7. Change current representative predictor immediately

8. Else if score of some predictor > THRESHOLD VALUE

9. Set this predictor as the representative predictor

10. End if

11. End if

12. For each predictor in Candidate Set

13. Do prediction and store the result

14. End for

15. Return the result of representative predictor

Step4: The iterative prediction

Since the classifier can predict only one value while quests are usually multi-step prediction[16], our algorithm creates an iterative approach. We invited length L data before the predicted point as the classifier input to get a prediction, and assumed the result as true to predict the next CPU load until all the steps had been finished.

V. MULTIDIMENSIONAL SPATIAL VOTING PREDICTION MODEL SUPPORTED BY GRAY PREDICTION MODEL

Resource performance prediction is becoming more and more important in cloud environment and CPU load prediction takes a great part in system maintenance and application schedule. The traditional method predicts future CPU load by one dimensional prediction. It works well in certain situations, such as on periodic servers. But there will be large errors in real-time prediction inevitably. The main reason is that the CPU load is not independent and relevant factors should be taken into account. So we proposed a multidimensional spatial voting prediction model supported by gray prediction model to solve this problem.

Forecasting process steps are as follows:

Step 1: Get sample data and divide the sample data into CPU load data and related factors (including memory usage, disk IO, network IO).

Step 2: Pre-process every related factor component X, $X = \{x_1, x_2, x_3, \cdots\cdots, x_n\}$, n represents the length of the sample. A new data sequence length n is obtain being processed by buffer operator once, denoted X_m wherein m represents X is processed data sequence treated by m-order buffer operator.

Step 3: Process each data sequence with 1-AGO to get new data sequence Y_m.

Step 4: Get the evaluation Z_m of the neighbors mean value of the new data sequence Y_m.

Step 5: Calculate the differential and derivative through the formula to get every order of parameters a and b in gray prediction model, resulting GM_m (1,1)[17], wherein m represents X whose data sequence was processed by m-order buffer operator.

Step 6: All of the gray prediction models are tested to measure MER[18] with the test data sample. Take the most accurate m order gray prediction model with the best MER. Calculate one dimensional prediction sequence of the three related factors data by it, denoted A_{mem}, A_{disk}, A_{net}.

Step 7: Construct training data set by three one dimensional predictive sequence and CPU historical load data.

Step 8: The four classifiers are trained to find the most optimized parameter of the training data set. Calculate the optimal parameters of individual particles and groups. Select the optimal parameters to construct four classifiers model.

Step 9: Vote for classifier model. Detailed pseudo code see (III.C). Choose the most suitable prediction model from the four classier models based on the voting results.

Step 10: Input the training data to calculate the final prediction value and iterate as needed.

VI. EXPERIMENTS

The experiment data in this paper comes from some company servers[9]. Data includes real CPU load(Fig.1) and three related variables(memory usage, disk IO, net IO)[10]. Sampling time is once per minute. The CPU load data is obvious regular or with little regularity, sometimes unusual.

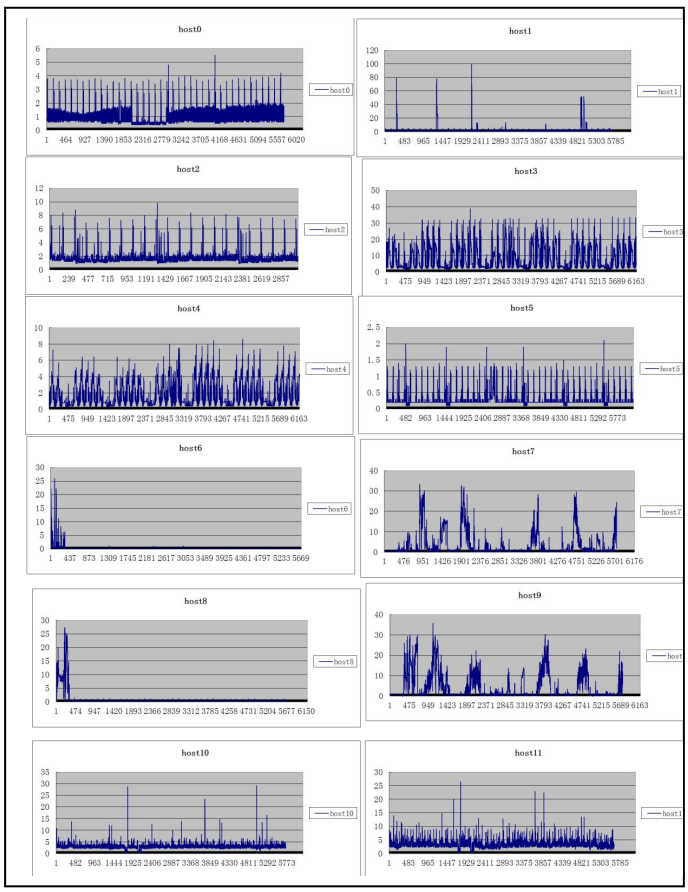

Fig. 1.CPU load data.

A. Compared with other algorithms to predict a single step

Three prediction model, including multidimensional spatial voting prediction model supported by gray prediction model mentioned in this article, proven ES algorithm and Similarity Prediction algorithm[19], have been used to do 10 one-step prediction. Data obtained are shown in Figure I. Errors are summed up as shown in Table I. We can clearly see though the table that the error of the multidimensional spatial voting prediction model is minimal.

TABLE I. ERRORS OF THE THREE ALGORITHMS

	Algorithm Name		
	multidimensional spatial voting	***ES***	***Similarity Prediction***
MER	0.5642	2.5476	1.4572
The maximum prediction error	0.9875	1.4265	1.3542
Mean square error	0.47341	0.9574	1.7567

TABLE II. PREDICTION SAMPLES

The actual value	***Predictive value***	***Error***	***Relative error***	***Accuracy /%***
multidimensional spatial voting prediction model				
20.41	20.51	0.1	0.05	99.5
12.26	11.88	-0.379	0.309	96.91
8.45	8.42	-0.03	0.036	99.64
2.21	1.211	-0.999	4.519	54.81
28.1	28.656	0.556	0.198	98.02
70.58	72.312	1.732	0.245	97.55
27.33	26.483	-0.847	0.331	96.9
2.72	2.156	-0.564	2.072	79.28
10.57	9.885	-0.685	0.648	93.52
48.3	46.348	-1.952	0.404	95.96
ES				
33.32	33.41	0.09	0.027	98.5
12.37	12.41	0.04	0.0323	94.5
3.57	3.62	0.05	0.1401	99.6
5.42	2.45	-2.97	5.4797	57.4
42.57	45.72	3.15	0.74	98.2
63.54	68.53	4.99	0.7853	48.5
35.74	38.75	3.01	0.8422	68.7
27.45	27.46	0.01	0.0036	95.4
17.86	15.42	-2.44	1.3662	86
45.72	48.75	3.03	0.6627	88

The actual value	*Predictive value*	*Error*	*Relative error*	*Accuracy /%*
Similarity Prediction				
24.21	24.56	0.35	0.1446	97
12.24	12.35	0.11	0.0899	99
8.41	8.82	0.41	0.4875	80
3.32	3.41	0.09	0.2711	88
27.5	27.45	-0.05	0.0182	99.8
70.8	71.42	0.62	0.0876	98
27.1	27.2	0.1	0.0369	99.7
5.4	5.5	0.1	0.1852	97.68
12.54	12.53	-0.01	0.008	99.9
48.7	48.8	0.1	0.0205	99.2

B. Compared with other algorithms to predict multi-step.

According to the previous results of the experiment, we made a prediction for multi-step by models including multidimensional spatial voting prediction model supported by gray prediction model mentioned in this article, proven ES algorithm, Similarity Prediction algorithm. Step length of 10,20,30,60,180 prediction was done for all data. Averaging the MER after testing the five sets of the experimental data. Except calculate MER, we also calculate BEST MER[20] to represent the best performance that each algorithm ever made. As the experimental data sampling period was 1 minute, prediction time were 10 minutes, 20 minutes, half an hour, one hour and three hours. The object is to test CPU load medium-term prediction. The following figures show the experimental results.

TABLE III. BEST-MER

steps	*multidimensional spatial voting*	*ES*	*Similarity Prediction*
10	1.4566	2.2345	2.6675
20	1.5789	2.4798	2.4234
30	1.8012	2.9436	2.9417
60	2.3235	3.2124	2.2789
180	2.9458	3.8323	2.335

TABLE IV. ALL MER

steps	*multidimensional spatial voting*	*ES*	*Similarity Prediction*
10	2.3456	3.4079	2.2376
20	2.9567	4.234	3.1958
30	3.9104	5.4671	4.4562
60	5.3234	6.2343	6.0356
180	12.457	13.3479	12.5231

The results show that the MER of multidimensional spatial voting prediction model is better than classic ES algorithm in all cases. Compared with Similarity Prediction algorithm in multi-step prediction, the result is very close. The BEST MER of our model is slightly inferior, but the gap is not great. According to Table IV, the MER of multidimensional spatial voting prediction model is basically the best, especially when the prediction steps is 144, our model is better than all of its opponent. What's more, the voting classifier algorithm is better than any single classifier algorithm when we take adaptability into consideration. Futhermore, though the results of multidimensional spatial voting prediction algorithm and Similarity Prediction algorithm are close, but on single point our model is superior.

VII. CONCLUSION

We present multidimensional spatial voting prediction algorithm model to break the limitation of one dimension prediction algorithm and single classifier prediction model. The future value of related data set is predicted by gray prediction model. We also trained all four classifier with the prediction result of related data set and historical data of CPU load to get the best parameters of each classifier, which are input for the prediction to choose the best classifier model by voting. Compared with classic models, the selected classifier model shows better suitability in complex real-time system, as the predicting results are more accurate with less MER and BEST MER, which means the multidimensional spatial voting prediction algorithm model we present is effective.

REFERENCES

[1] S. Ghosh. Distributed Systems: An Algorithmic Approach. Chapman & Hall. 2006.

[2] P.A. Dinda. The statistical properties of host load. Scientific Programming 7(3,4). 1999.

[3] P.A. Dinda, D.R. O' Hallaron. Host load prediction using linear models. Cluster Computing 3(4), 2000: 265 - 280.

[4] R. Wolski, N. Spring, J. Hayes. Predicting the CPU availability of time-shared unix systems on the computational grid. In Proc. 8th IEEE Symp. on High Performance Distributed Computing. 1999.

[5] P. A. Dinda, D. O' Hallaron. Realistic CPU Workloads Through Host Load Trace Playback. In Proc. 5th Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers, Rochester, NT. May, 2000.

[6] R. Wolski. Dynamically Forecasting Network Performance Using the Network Weather Service. Journal of Cluster Computing. 1998.

[7] R. Wolski, N. Spring, and C. Peterson. Implementing a Performance Forecasting System for Metacomputing: The Network Weather Service. Proceedings of SC97. 1998.

[8] R. Wolski, N.T. Spring, and J. Hayes. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. Future Generations of Computer Systems. 1999.

[9] P.A. Dinda, D.R. O' Hallaron. An Evaluation of Linear Models for Host Load Prediction. In Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing (HPDC' 99) (August 1999). 1999: 87 - 96.

[10] L. Yang, I. Foster, and J.M. Schopf. Homeostatic and Tendency-based CPU Load Predictions. Int' l Parallel and Distributed Processing Symp. (IPDPS' 03). 2003: 42 - 50.

[11] Y. Zhang, W. Sun, and Y. Inoguchi. CPU Load Predicitons on the Computational Grid. IEICE Trans. Inf. & Syst., Vol. E90-D, No. 1. January 2007.

[12] Y. Zhang, W. Sun, and Y. Inoguchi. Predict task running time in grid environments based on CPU load predictions. Future Generation Computer Systems Volume 24, Issue 6. July 2008: 489 - 497.

[13] K.B. Bey, F. Benhammadi, A. Mokhtari, and Z. Guessoum. CPU Load Prediction Model for Distributed Computing. 2009 Eighth International Symposium on Parallel and Distributed Computing. 2009.

[14] K.B. Bey, F. Benhammadi, A. Mokhtari, and Z. Guessoum. Mixture of ANFIS systems for CPU load prediction in metacomputing environment. Future Generation Computer Systems, Volume 26 Issue 7. July 2010: 1003 - 1011.

[15] J.S. Jang. ANFIS: Adaptive-network-based fuzzy inference systems. IEEE Trans. on Systems, Man, and Cybernetics, 23(03). May 1993: 665 - 685.

[16] M. Swany and R. Wolski. Multivariate Resource Performance Forecasting in the Network Weather Service. In Supercomputing’02. 2002.

[17] S. Vazhkudai, J. Schopf. Predicting sporadic grid data transfers. In the 11th IEEE Symposium on High Performance Distributed Computing. 2002.

[18] S. Vazhkudai, J. Schopf. Using disk throughput data in predictions of end-to-end grid data transfers. In the 3rd International Workshop on Grid Computing (GRID 2002). November 2002.

[19] J. Liang, K. Nahrstedt, and Y. Zhou. Adaptive Multi-Resource Prediction in Distributed Resource Sharing Environment. 2004 IEEE Int’l Symp. on Cluster Computing and the Grid. 2004: 1 - 8.

[20] S. Akioka and Y. Muraoka. Extended Forecast of CPU and Network Load on Computational Grid. Fourth IEEE International Symposium on Cluster Computing and the Grid (CCGrid’04). 2004.

[21] M. Swany and R. Wolski. Multivariate Resource Performance Forecasting in the Network Weather Service. In Supercomputing’02. 2002.

[22] S. Vazhkudai, J. Schopf. Predicting sporadic grid data transfers. In the 11th IEEE Symposium on High Performance Distributed Computing. 2002.

[23] S. Vazhkudai, J. Schopf. Using disk throughput data in predictions of end-to-end grid data transfers. In the 3rd International Workshop on Grid Computing (GRID 2002). November 2002.

[24] J. Liang, K. Nahrstedt, and Y. Zhou. Adaptive Multi-Resource Prediction in Distributed Resource Sharing Environment. 2004 IEEE Int’l Symp. on Cluster Computing and the Grid. 2004: 1 - 8.