An Idle Compute Cycle Prediction Service for Computational Grids

Suntae Hwang[1], Eun-Jin Im[1], Karpjoo Jeong[2], and Hyoungwoo Park[3]

[1] School of Computer Science, Kookmin University, Seoul, Korea
{sthwang, ejim}@kookmin.ac.kr
[2] College of Information and Communication, Konkuk University, Seoul, Korea
jeongk@konkuk.ac.kr
[3] Supercomputing Center, KISTI, Daejon, Korea

Abstract. The utilization of idle compute cycles has been known as most promising and cost-effective way to build a large scale high performance computing system, but not widely used because of the lack of effective idleness prediction techniques. In this paper, we argue PCs at university computer labs have a great potential for the utilization of idle CPU cycles, and propose two techniques for predicting idle cycles of those PCs: heuristic and statistical. Based on these techniques, we present the design and implementation of an idle compute cycle prediction service for computational grids. Our experimental results show that the utilization of idle compute cycles is a viable approach to cost-effective large scale computational grids.

1 Introduction

Scientific applications such as Molecular Simulation, High Energy Physics and Genome Informatics have challenging requirements for computation which can not be satisfied by the conventional supercomputing technology in the near future. Recent technical advances in grid computing provide us with an opportunity to tackle those problems by aggregating a large number of computers at organizations around world [3,6,7]. Finding idle resources is crucial for such approach; otherwise, organizations have to invest new computing platforms for grid computing. However, to predict the idleness of computing resources is generally very difficult because the schedulers do not have full knowledge about future tasks or take full control over them.

In this paper, we argue that university computer labs have a great potential for grid computing platforms for the following reasons. First, university computer labs have hundreds or thousands of PCs(Personal Computers) whose aggregate computing power can equal that of supercomputers. Second, those PCs are idle most of the time (e.g., at night). Moreover, they are almost completely idle during vacations. Third, PCs that do not have owners or valuable data are much less sensitive to the privacy issue that other computing resources. Finally, students' PC usage patterns are more regular and predictable than usage patterns of other computing resources because class schedules and school administration

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3036, pp. 116–123, 2004.

schedules which are often repeatable and predictable determine students' activities in schools to a large extent[5].

In this paper, we propose two techniques for predicting idle compute cycles of PCs at university computer labs: *heuristic* and *statistical*. The heuristic technique uses the usage patterns of the last week for predicting those of the current week, but the statistical technique uses the *Hidden Markov Model* (HMM) [1] to predict future idle cycles. We compare these two techniques by real monitoring data collected for six months. Based on these two techniques, we present the design and implementation of an idle cycle prediction service for computational grids.

2 Idle Compute Cycle Prediction Models for University Computer Labs

Before we go on to data collection and analysis, first we clarify assumptions on our university PC labs.

- We do not consider the remote login or job queue length in both monitoring systems, because Microsoft Windows 2000 Professionals are running on the PCs. This is strict, but in this way, we can be sure that the user is not disturbed by other users harvesting his idle cycles. That means Grid jobs are submitted to those PCs by special scheduler, while other users can not submit remote jobs to those PCs.
- The PC rooms are classified into two types. One type of rooms, which we call *lecture rooms*, are assigned to school classes. The other type of rooms, which we call *open rooms*, are always open to students during office hours. Computers in *open rooms* are freely accessible to the users with authorized cards, while computers in *lecture rooms* are exclusively used by the enrolled students during lecture time but also freely accessible to the users between lecture times. Open hours are from 9AM to 9PM for both type of rooms.
- In this analysis, we treat one hour as a basic unit of idle-ness; that is, although a PC is used for a short period of time, it is assumed to be busy for the entire hour.

2.1 Data Collection

These days, most of PC rooms in universities adopt card system, where individual PC in the facility has magnetic card reader and the user gets authorization with their card to get access to a computer. The card system keeps track of data about when a PC is occupied by a user.

In our institution, we have collected card system log for the open rooms, but the card system log was not available for the lecture rooms, because the card system is off when the room is used for the lecture. For that reason, we ran a simple program on PCs in our lecture room; it infinitely repeats collecting load information in one minute's interval. We collected approximately 9.2MB of data on 54 PCs from April 2002 to September 2002. The raw monitoring

data is converted into a sequence of idle, busy, and unknown periods. In this latter monitoring system for lecture rooms, a computer is defined to be idle if the keyboard or the mouse has not been touched for the last minute; otherwise, it is busy. In the card system for the open rooms, the definition of idle state is slightly different, where it means the previous user logged out, and none has logged in yet.

Because our primary concern was long running jobs, the data is again converted at hourly interval, where the state of a computer is marked *busy* if it was marked busy at least one minute in the hour, and marked *idle* otherwise. Since unknown state is the case when the PC is turned off, we marked unknown state idle because all PCs can be actually used if the administration policy is changed to turn on all PCs for 24 hours.

2.2 A Weekly Usage Pattern-Based Model and a HMM-Based Model

First, we designed a heuristic model based on our intuition that students' PC usage patterns are largely affected by class and school administration schedules usually repeated in a weekly basis. In this model, we simply assume that idle-ness patterns are repeated in a weekly basis. That is, we predict the idle-ness patterns of the current week based on those of the previous two weeks. For example, we predict a PC will be idle from 1:00pm to 2:00pm this Tuesday, only if the PC was idle during the same period of time last Tuesday and the previous Tuesday. For this reason, we call this model a weekly usage pattern-based or schedule-based model.

We also modeled the usage pattern of PCs using a *Hidden Markov Model* (HMM) In this model, we build a HMM for each PC in computer rooms, using two week's history of the computer usage. The number of states in the HMM was arbitrarily chosen to be 10, since, after testing different number of states, we concluded that the accuracy of the prediction was not much different in HMM with larger number of states. Then in the prediction step, we generate a most probable sequence of length equal to one week's usage sequence, using a Viterbi algorithm [8]. The generated sequence is used as a predicted usage pattern for the next week. The sequence length was in the unit of weeks, based on the observation in the previous subsection. With two week's usage pattern, we expect the HMM learns the hidden states of weekly pattern.

2.3 Comparison of Two Prediction Models

We compared the effectiveness of the two prediction models for various circumstances. We calculated *hit ratios* (ratios of correct predictions) of both prediction models for each PC on real monitoring data.

In this experiment, we assumed that the job length is two hours long because we are interested in long-running jobs. Each prediction is either hit (correct) or miss (wrong). We call a miss failure. Failures are again classified into two categories: *critical* and *non-critical*. The critical failure is the case where the

model predicts the busy state of PC in the real world to be idle. The non-critical failure is vice versa. The former is more serious than the latter because that prediction causes a busy computer to be interrupted and scheduling work to be wasted.

The figures 1, 2 and 3 compare the hit ratios and critical failure ratios of the two prediction models. The prediction was again tested in three modes. In the first mode shown in figure 1, the prediction was performed using the whole data. In the second and third modes shown in figures 2 and 3, the prediction was performed respectively using the data for the daytime (9AM-9PM) and for the nighttime (9PM-9AM). For the schedule-based heuristic model, it only means that the times of the day of collected data are different, but for the statistical model based on HMM, it also means that the constructed HMM in each of three modes are different, because the training data were different. That is, the day model was built only with history of the daytime in the previous two weeks.

The left graphs in the figures shows the hit ratio, and the right graphs are for the critical failures. The data are again collected for different period, first two groups of bars for the whole 21 weeks excluding the first two weeks, the next two groups for the first 7 weeks (weeks 2-9) which belongs to the spring semester, and the final two groups for the remaining weeks in summer vacation. The bars in the graphs are paired for the two different prediction models, and the pairs of two different type of rooms, open room and lecture room, are again shown next to each other. For each pair of bars, the usage rate of the corresponding period is shown on the top of bars, so that we can relate the effectiveness of the models to the usage rate of computers.

Overall, the hit ratios are comparable in the two prediction models, but as the usage rate of the computer increases, the accuracy of the HMM-based model tends to be worse than that of the heuristic model. More importantly, we note that the rate of critical failures of the heuristic model is consistently lower than that of the HMM-based model.

3 Idle Compute Cycle Prediction Service

A computation grid consists of various services. To utilize idle compute cycles in university computing room for computational grid, we need local resource management system that can continuously update PC's load state in computing room, and monitor and control allocated jobs.

Applications designed to execute on computational grids frequently require the simultaneous co-allocation of multiple resources in order to meet performance requirements. Since this paper focuses on utilizing of idle compute cycles, we mention to computational resources only although the definition of resources includes all devices that an application might require, including networks, memory, storage, rendering hardware, and display devices. Because it is not reasonable to assume a centralized global scheduler in Grid environment, scheduling conflicts may result in over all degraded performance. One approach to enhance the local resource management system is to incorporate advance reservation capabilities

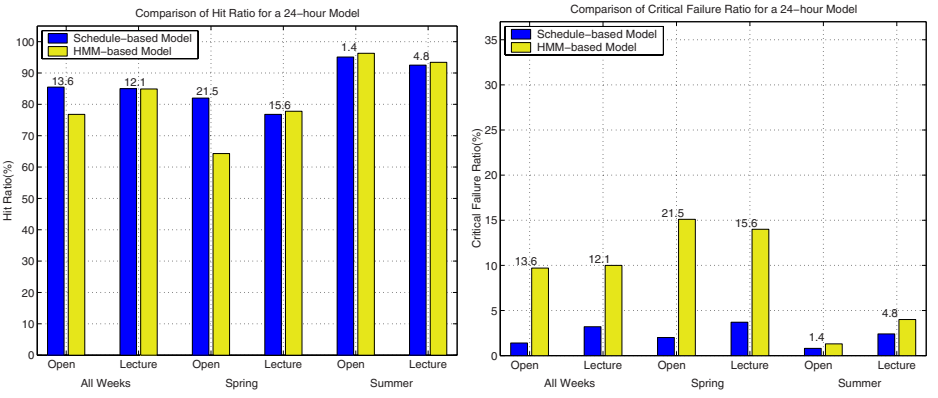

Fig. 1. Prediction Result of two Models in a 24-hour mode. The numbers on the top of bars are the usage rates of the corresponding period.

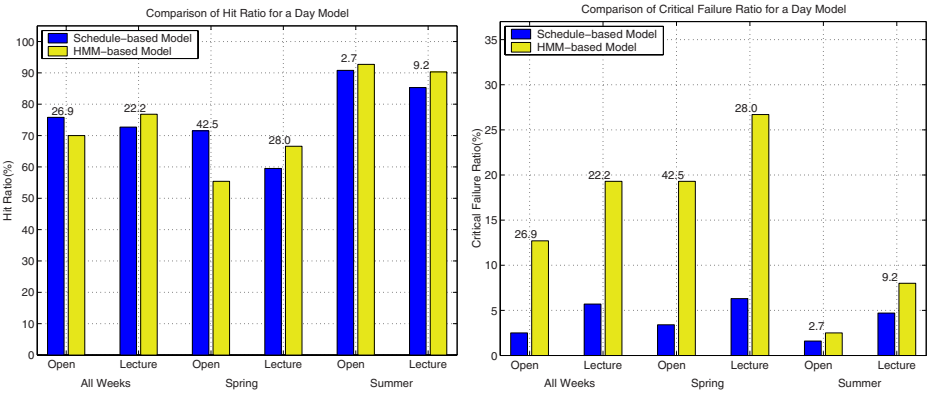

Fig. 2. Prediction Result of two Models in a day mode.

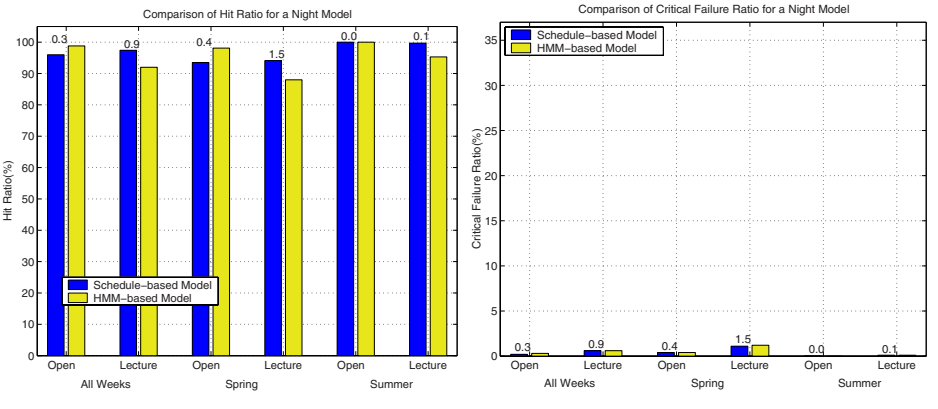

Fig. 3. Prediction Result of two Models in a night mode.

into a local resource manager. Then, a co-allocator can obtain guarantees that a resource will deliver a required level of service when required[2,4].

Therefore forecasting of the load of computational resource is very important for resource reservation and co-allocation. Therefore, local resource management system needs to provide continuously prediction service. Grid system on OGSA foundation is formed in various service form. Among these, scheduler or co-allocator will also exist as OGSA service form. To properly operate such services, local resource state should be predicted by prediction service. And based on this, resource will be either allocated or reserved.

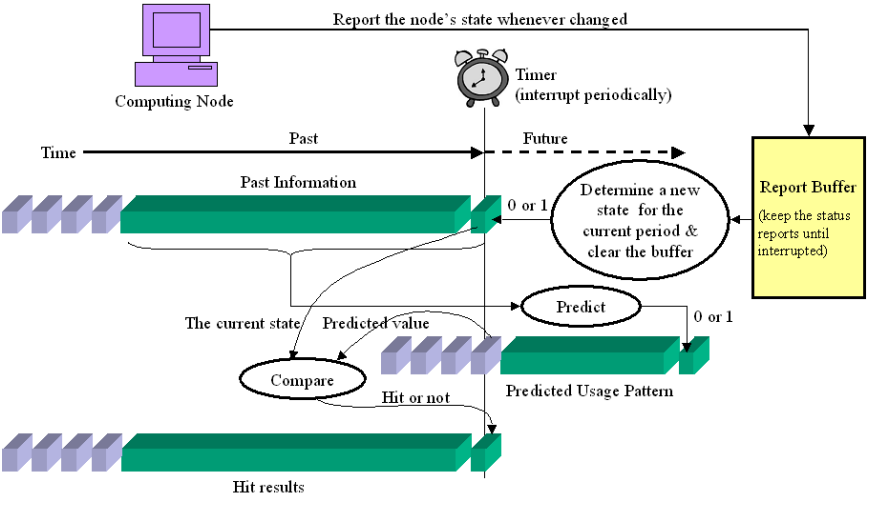

Fig. 4. Prediction service implementation. Whenever timer periodically interrupts, circled actions will be processed. Here, we have used schedule-based prediction model which was described in Section 2.

4 Implementation

Based on prediction model, which was described in Section 2, we have implemented and added prediction service to our local resource management system. States of nodes that are reported from each PC in compute room, are recorded in resource management system's buffer. And current period's state is determined whenever periodical (e.g. one hour) interruption occurs based on the state reports which were recorded in buffer. In other words, it is *idle* only reported states in buffer are all idle, otherwise it is regarded as *busy* if there is even one busy state was reported during the interval. As shown in Figure 4, if timer interrupts, prediction server determines the new state of the node, predicts a state of a week ahead according to the updated past information, and recalculates the usage ratio and hit ratio of the given duration(for example two weeks in our case).

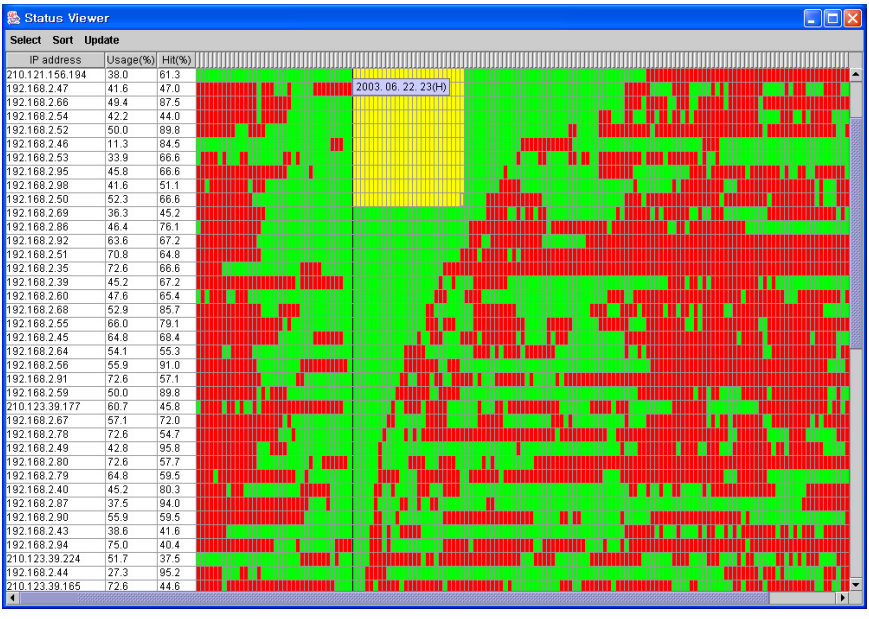

Fig. 5. A resource allocation tool using the idle cycle prediction service. The first column represents host IP address and the second and the third represent usage rate and hit ratio of the node respectively. The rest represent the predicted state of the node for a week ahead. Small dark boxes are the ones predicted as *busy* and the rest are the ones predicted as *idle*. If you put mouse on the head row, tool tip, which shows applicable date and time, will show. When you click the mouse button on the tool tip, data will be sorted in order which is predicted as will be idle for longest period based on that moment(indicated by the vertical line). And little more lightly marked block is selected part for reservation.

Figure 4 shows a data structure for a single node. The prediction server returns predict patterns, host IP addresses, usage ratios and hit ratios as many as number of nodes. Figure 5 shows a snapshot of a tool which can be used together with job preparation tool.

5 Conclusions and Future Work

The lack of effective techniques for predicting long idle cycles is one of major obstacles to building a large scale cost-effective high performance computational grid. In this paper, we proposed two techniques to predict long idle cycles for PCs at university computer labs. One technique is designed based on our intuition that students' PC usage patterns are largely affected by class and school administration schedules usually repeated in a weekly basis. The other technique is based on a statistical model called Hidden Markov Model (HMM).

We examined the effectiveness of these two techniques based on real monitoring data for PCs at our university computer labs. For these experiments, PCs were monitored for about six months. Overall, both of these two techniques consistently showed high correct prediction rates for various circumstances. But our intuition-based heuristic technique outperforms the HMM-based technique in most cases. For the crucial case where busy states are predicted to be idle, the effectiveness of the heuristic technique is significantly better than that of the HMM-based technique. That is, our intuition about idleness patterns of PCs is proven to be better.

Based on these techniques, we presented the design and prototype implementation of a idle cycle prediction service for computational grids. Currently, we have been integrating this service into Globus toolkit 2.0. In addition, we plan to reimplement this as a service for Globus toolkit 3.0 and to develop a grid meta scheduling technique based on this service.

Acknowledgments. This work was partially supported by MIC (Ministry of Information and Communication) through National Grid Infrastructure Implementation Project of KISTI (Korea Institute of Science and Technology Information). It was also in part supported by University IT Research Center Project and by Grant No. R04-2002-000-20066-0 from Korea Science and Engineering Foundation.

References

1. Jeff Bilmes. What HMMs Can Do. UWEE Technical Report UWEETR-2002-0003, University of Washington, January 2002.
2. Karl Czajkowski, Ian Foster, and Carl Kesselman. Resource co-allocation in computational grids. In *The Eighth IEEE International Symposium on High Performance Distributed Computing*, August 1999.
3. I. Foster and C. Kesselman. Globus: A Toolkit-based Grid Architecture. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 259–278. Morgan Kaufmann, 1999.
4. I. Foster, C. Kesselman, C. Lee, R. Lindell, and K. Nahrstedt. A distributed resource management architecture that supports advance reservations and co-allocation. In *International Workshop on Quality of Service*, 1999.
5. Suntae Hwang, Karpjoo Jeong, Eun-Jin Im, Chongwoo Woo, Kwang-Soo Hahn, Moonhae Kim, and Sangsan Lee. An Analysis of Idle CPU Cycles at University Computer Labs. *Lecture Notes in Computer Science*, 2667(1):733–741, 2003.
6. C. Liu, L. Yang, I. Foster, and D. Angulo. Design and evaluation of a resource selection framework for Grid applications. In *Proc. of 11th IEEE Symposium on High Performance Distributed Computing*, July 2002.
7. V. Subramani, R. Kettimuthu, S. Srinivasan, and P. Sadayappan. Distributed job scheduling on computational grids using multiple simultaneous requests. In *Proc. of 11th IEEE Symp. On High Performance Distributed Computing*, july 2002.
8. A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Informat. Theory*, IT-13:260–269, April 1967.