Finding Idle Periods on Networks of Workstations

NYU Computer Science Dept. Technical Report 761

Peter Wyckoff
Computer Science Department
New York University

Theodore Johnson
AT & T Research

Karpjoo Jeong
Computer Science Department
Kon-Kuk University

March 4, 1998

Abstract

We present a simple technique for predicting the probability that an idle workstation will continue to be idle for i minutes, given that it has been idle for x minutes (i.e., find the *remaining idle period probability* $P(i;x)$). By idle we mean that the workstation owner is not interactively using the workstation or executing other tasks on it. The results are particularly applicable to the scheduling of tasks in systems that harvest cycles from idle-only workstations.

Our Remaining Idle Period Probability Predictor (RIPPP) uses the distribution of the lengths of idle periods on the managed workstations. Collecting, storing, and processing these distributions (in the form of histograms) is a small overhead on modern workstations (a few kilobytes of storage per workstation).

We investigated the behavior of our RIPPP with usage traces of 31 workstations collected over a five month period, and discovered the following six results. (1) The distribution of one month of idle periods predicts the remaining idle period probability in the next month for most workstations. (2) Different workstations tend to have significantly different idle period length distributions. (3) The average length of an idle period does not necessarily correlate well with the probability of being able to find long idle periods, contrary to intuition and previous scheduling heuristics. (4) A workstation that has been idle a long time does not necessarily have a high probability of remaining idle for a long time. (5) Using the time of day can improve predictions. (6) The length of the previous and the current idle periods are positively correlated, but the length of the previous idle period is not strongly correlated with finding long remaining idle periods.

Based on these studies, we conclude that an effective way to find idle workstations is to collect their idle period length distribution and use it to compute $P(i;x)$. We believe our analysis will be applicable to predicting the length of busy periods, which is useful for deciding whether to migrate or suspend tasks when a workstation becomes busy (the owner reclaims it).

From our results, we have developed a remaining idle period probability toolkit which includes a statistics collector and a prediction library in C. This will be available from our project homepage.

1 Introduction

Harvesting the idle cycles of a network of workstations is attractive for many reasons: low priority batch sequential tasks such as index rebuilds for a database, high priority batch sequential tasks such as parallel make, and background parallel tasks. Generally, networks of workstations are underutilized [16, 22, 10, 11, 15, 8, 9]. There are many systems designed to utilize the aggregate power of a network of workstations [13, 14, 21, 9, 20, 18, 19, 17]. Condor and Piranha are specifically designed with the goal of being as unobtrusive as possible to users. In the Condor system, when a workstation becomes busy, the system immediately checkpoints any remote tasks running on the workstation and then uses the checkpoint file to start the tasks on idle workstations. In the Piranha system, the programmer is responsible for writing a *retreat function* which is invoked when a workstation upon which the program is running becomes busy.

Systems such as these, which use a non-dedicated network of workstations, must include a scheduler that decides where to place new tasks and when to migrate tasks from busy or overloaded workstations to idle, under-loaded workstations. The scheduler can benefit from knowing the length of incoming tasks, and the duration of idle and busy periods. We address the latter issue by offering a means of predicting the remaining idle (busy) period probability. For many scientific applications, for example, it is the case that the approximate task length is known.

In this paper, we use the distribution of the lengths of idle periods on workstations to create the *Remaining Idle Period Probability Predictor* (RIPPP) — the probability that an idle workstation will continue to be idle for i minutes, given that it has been idle for x minutes, $P(i;x)$. We do so by creating histograms from each workstation trace and then using these create the RIPPP. A drawback of this method is that it requires that an empirical distribution be collected and the evaluation of a more complicated function than just keeping the expected idle period length. The amount of statistical information to be kept is $(Idle_{max}/BinSize) * sizeof(short)$ (a few kilobytes in total), the statistics collection daemon runs for a few milliseconds every minute, and the idleness predictor requires a few additions and multiplications. On today's processors, these costs are negligible.

Simple heuristics such as using the expected idle period length to predict idle periods do not work well. An advantage of this approach is that only one number needs to be used per workstation — the expected idle period length. The advantage of collecting statistical information is that we can better predict which workstation has the highest probability of accommodating a task of a known length. For example, if one workstation has the characteristic that it is idle for 20 minutes for half the idle periods and 30 minutes for the other half, and another workstation is idle for 10 minutes for half the idle periods and 2 hours for the other half, the latter workstation will have a higher expected idle period length. Yet if both workstations have been idle for 2 minutes, it would be better to place a 15 minute task on the first workstation. As we will see in the experimental results, this type of situation occurs often in practice.

Once a task has been scheduled on a workstation, what happens if the workstation becomes non-idle before the task completes? There are three choices: (1) kill the task and restart it elsewhere if it is idempotent and or fault tolerant, (2) suspend the task in the hope that the workstation will become non-idle in the near future, or (3) migrate the task to another workstation. For the scheduler to effectively decide among these choices, it must not only be able to predict the length of idle periods on other workstations in the workstation pool, but also the length of the busy period on the workstation that has become busy. For this, the remaining busy period probability is a good measure.

Another application of our predictor is choosing between fault tolerance methods. If we model non-idleness as failure (to be as non-obtrusive as possible to users returning to their workstations by immediately killing remote tasks) fault tolerance is a necessity. In this setting, it may be that different fault tolerant choices are available. For example, checkpoint every minute or every ten minutes. If a workstation is likely to be reclaimed in the near future, a fault tolerance method that has low recovery time is appropriate. If the workstation is likely to remain idle for a long time, a low runtime overhead fault tolerance method is more appropriate. Since RIPPP gives us the probability that a sufficiently long idle period exists, our idleness predictor can be used to predict the expected running time of a process under different fault tolerance methods and then the method that minimizes overall running time can be chosen.

We test the effectiveness of the remaining idle period probability function using five month traces for a pool of 31 workstations, using them to create empirical distributions of the idle period length. We need an assurance that the past does predict the future, because we are using a statistical profile to predict future events. We develop a test for the traces that we have collected that is appropriate for our application. This test, which computes an average prediction error, shows that the idle period length distribution of one month can be used to predict the remaining idle period lengths on the next month on the same workstation.

Our basic predictor uses only the current length of the current idle period to predict the remaining idle period probability, RIPPP, $(P(i;x))$. We experiment with using time of day and the length of the previous idle period to make our predictions more precise. We show that including time of day information improves the accuracy of the RIPPP.

In addition to validating our idle period length probability predictors, our results show information about the actual dynamics of idle periods. For example, a workstation that has been idle for a long time is not necessarily likely to remain idle for a long time, and workstations with long average idle periods are not always likely to have very long idle periods.

The remainder of this paper is organized as follows. In the next section we present related work. In section 3, we define the terms used throughout the paper. In section 4, we discuss our data collection techniques and idleness criteria. In sections 5 and 6, we present our basic idleness predictor and our metric for comparing predictions. We then present our experimental results. Section 9 briefly discusses the remaining idleness probability prediction library. Section 10 summarizes the paper.

2 Related Work

In [1], Golding et al. investigate finding idle periods on disk drives for performing maintenance tasks. They present a number of fast prediction algorithms. They found that fixed duration, moving average, and adaptive backoff algorithms worked well. Due to the prevalence of very short idle periods, these methods are not appropriate for predicting the lengths of idle periods on networks of workstations.

In [2] Acharya, Edjlali, and Saltz investigate finding large pools of idle workstations for parallel computation. They suggest using the *idleness cutoff*, the median idle period length, to characterize each workstation. They found this value to be high and therefore suggest that any workstation that is idle for more than five minutes is an equally good candidate for task placement. Our method is more precise and we found that having one rule (such as waiting five minutes) for all workstations does not work well. For the workstations in their study they found that their idleness behavior was different from one another. Our results agree.

In [3] Mutka and Livny studied workstation availability patterns and fit the available period length distribution to a three stage hyperexponential distribution. They find that the lengths of idle periods tend to be correlated. We find that the idle period length distributions have important features that would not be captured by a fitted model. We also find that the length of the previous idle period does not provide good information on whether the current idle period has a long remaining life.

In [4] Samadani and Kaltofen develop a time series model of current system load. In [5] Harchol-Balter and Downey analyze process lifetimes and develop an analytical model for use in process migration algorithms.

In this paper we develop a precise estimator of the residual life of the current idle period in a workstation. Because we base our RIPP predictor on an empirical distribution, we do not need to rely on simple heuristics (which are often inaccurate). Furthermore, the RIPP predictor returns the probability that a workstation will remain idle for the desired length of time, and this information is can be used to choose between recovery methods.

3 Definitions

- idle period: a period on a workstation in which the owner is not interactively or otherwise using the workstation.
- busy period: a non-idle period of a workstation.
- *remaining idle period probability* $P(i;x)$: the probability that a workstation that has been idle for i minutes will remain idle for another x minutes.

- CDF: Cumulative Distribution Function.
- RIPPP: Remaining Idle Period Probability Predictor.

4 Data Collection

We ran a Perl daemon on each workstation; it looped continually, collecting load information and then sleeping for a minute. The load information we collected was the average number of jobs in the run queue over the last 1, 5, and 15 minutes (using the Unix uptime command), the last times the mouse, keyboard, and console were used, and the last time a remote logon occurred. We collected 344 MB of data on 43 workstations from December 1996 to May 1997. 12 of the workstations were excluded from the analysis because they were removed from the network for other experiments a large portion of the 5 month period. There are many holes in the data for these workstations, and thus, there are not as many sample data points making the data less statistically significant. For the 31 workstations used in our analysis, on average there are $8,270$ idle sample points and $8,362$ busy sample points.

On the raw data, we ran a conversion script which generated a sequence of idle, busy, and unknown markers (for periods when the workstation was down or our daemon was not running), each with a time of day and length attribute.

A workstation was idle if the following criteria were met, otherwise it was considered busy: (1) the console, keyboard, and mouse had not been touched for the last minute, (2) the remote logon record file had not been updated for the last minute (this was more strict than necessary since logouts cause the file to be updated), and (3)the average number of jobs in the runnable queue over the last minute, 5 minutes, and 15 minutes was less than or equal to 0.1 , 0.08 , and 0.03 , respectively.

Our criteria is strict, but in this way we can be sure that the user is not disturbed by other users harvesting his idle cycles. The main possible disturbance is remote tasks using large amounts of physical memory thereby swapping out the memory pages of the owner's suspended processes; this can be avoided by restricting the amount of physical memory available to remote tasks.

5 Remaining Idle Time Prediction

To make things precise, we describe the method of determining the probability that a workstation is idle for another i minutes given that it has been idle for x minutes. We denote this value by $P(i;x)$, the *remaining idle period probability.* This quantity is the survivor function of the excess life of the idle period. Let X be a random variable representing the count-averaged length of idle periods at a work station (i.e., the length of each idle period is a sample from X). Let F and f be the CDF and the PDF of X, respectively.

Suppose that we query a workstation and find that it is idle. Then, the (time-averaged) probability that the idle period is of length s is $Ksf(s)ds$, where $K = 1/\overline{F}$ [6]. But suppose that we know that the workstation has been idle for x minutes when we performed the query. The probability of arriving x minutes into an idle period of length s is x/s if $x < s$ and 0 otherwise. Let $f'(s;x)$ be the PDF of the probability arriving x minutes into a s minutes idle period at the workstation. Then,

$$f'(s;x) = \begin{array}{ll} K'f(s) & x < s \\ 0 & \text{otherwise} \end{array}$$

Where $K' = 1 - F(x)$. Let $P(i;x)$ be the probability that the current idle period will last for another i minutes, given that it has lasted for x minutes so far. Then

$$P(i;x) = \frac{1 - F(i+x)}{1 - F(x)}$$

To build a RIPP that estimates $P(i;x)$, we estimate $F(s)$ by collecting histograms of idle period lengths. The histogram information is compact. Even taking fine-grained histograms with a bucket resolution of one minute, only 2880 buckets are required to record idle periods of up to two days in length. Using short ints to store bucket occupancies, the space requirement is less than 6 Kbytes per histogram even without using compression techniques. For modern workstations, this is a minor storage requirement. The computation of F needs to be done when histogram collection is finished. At run time, only two array lookups and one division are required to estimate $P(i;x)$.

6 Comparing Predictions

In many of the experiments, we compare one set of $P(i;x)$ predictions with another set. Furthermore, we need to perform this for a large set of workstations and present summarized results.

Standard statistical tests for comparing distributions are not appropriate for our purposes. For example, the Kolmogorov-Smirnov test uses the maximum difference in the CDF of a distribution. When we applied this test to idle period length distributions, we would find that two idle period length distributions that gave almost identical remaining idle time predictions would be judged significantly different by the KS test. For example, one distribution would have significantly more very short idle periods than the other, but would show the same tail behavior, and $P(i;x)$ is primarily influenced by the tails.

The comparison method that we decided on is the average difference in absolute value between the predicted probability of the remaining idle period lasting for more than i minutes. We took this average over all histogram boundary points (we used two minute buckets) until one of the CDFs that the predictions are based on reached 0.995. To ensure that we are not biasing the comparison by choosing a particularly 'good' or 'bad' value of i, we typically ranged i between 4 and 40 minutes in steps of 2 minutes.

Suppose that we have computed $P(i;x)$ and $P'(i;x)$ for $i \in \{i_1, \ldots, i_S\}$ and $x \in \{x_1, \ldots, x_R\}$ Then we compute the *average difference* between P and P' to be

$$diff(P, P') = \sum_{s=1}^{S} \sum_{r=1}^{R} |P(i_s; x_r) - P'(i_s; x_r)|/R * S$$

The comparison method we chose is similar to computing the mean squared error in the RIPPPs, however, it has the advantage of a more readily interpretable meaning.

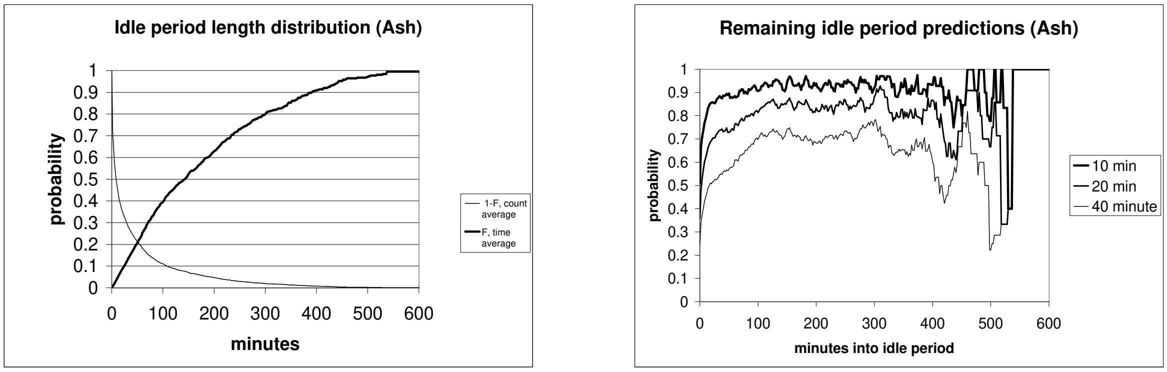
7 Idle Period Length Predictions

In this section, we present remaining idle period predictor results for three representative workstations. One of the workstations is a private workstation, and the other two are public workstations.

In Figure 1 part (a), we show the curve for the survivor function $(1-F)$ of the count-averaged idle period length distribution, and the CDF of the time-averaged idle period distribution for the private workstation Ash. Most idle periods are short, and thus, the remainder of the CDF curve drops off rapidly. However, if one samples the workstation and finds that it is idle, one is likely to find it in a long idle period (e.g., a 45% chance of the idle period being 200 minutes or longer). Part (b) shows the probability that the remaining life of the idle period is 10, 20, and 40 minutes, given that the workstation is x minutes into an idle period at the time of the sampling. This workstation is a good candidate (70% probability of success) for placing 20 minutes jobs if the workstation has been idle for 23 minutes or longer.

In Figure 2, we show the same data for Wilma, a public workstation. We have blown up the $1-F$ curve for the count-averaged idle period length distribution to focus on a "flat spot" in the CDF. This flat spot represents an opportunity for job placement, because the failure rate of idle periods is low in this region. In part (b), we see that one is almost guaranteed to find 10, 20, and 40 minute remaining idle periods if the current idle period is in the flat spot (between 250 and 500 minutes).

In Figure 3 we show the same data for Doc, another public workstation. Doc has a very different idle period length distribution distribution than Wilma, and in fact one is not likely to find long remaining idle periods on this workstation. Note that the shape of the distribution is similar for Doc and Ash, yet they give very different predictions. We also note that the average length of an idle period on Doc is 8.2 minutes, while the average length of an idle period on Wilma is 28 minutes. However, one is likely to find remaining idle periods of 40 minutes or longer on Wilma, but one is unlikely to find remaining idle periods of 10 minutes



(a) (b)

Figure 1: (a) Idle period length distribution and (b) remaining idle period prediction for Ash. In part a of the figure, notice that although there are many short idle periods, there is a 45% chance being in a 200 minute or longer idle period if the workstations idle periods are sampled randomly. In part b notice that after 23 minutes the probability that the idle period will last another 20 or more minutes is 70 %.

on Doc. Hence we see a demonstration that the average length of an idle period is not a good predictor of the ability to find long idle periods.

8 Experiments

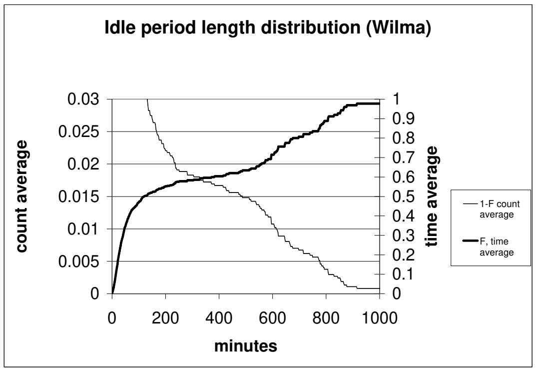
In this section, we discuss our experiments for validating the applicability of our remaining idle period predictor, and for determining what other information is useful in predicting future idle period lengths.

8.1 Does the Past Predict the Future?

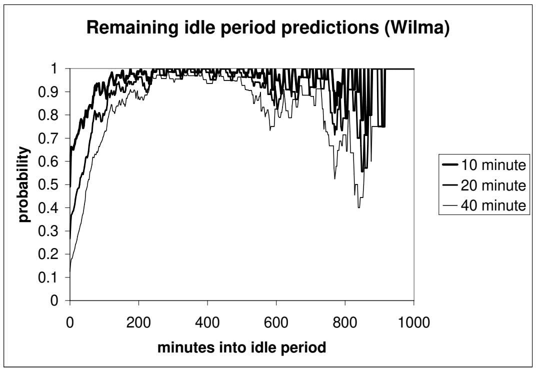
Our remaining idle period predictor is based on collecting the idle period length distribution, and then using it to make predictions. For this technique to work, we need assurance that a workstation's behavior is stable over time.

We tested the ability of the past to predict the future by partitioning the idle period trace of each workstation into two successive periods. We computed the average difference between the past and future RIPPPs using the method discussed in section 6. We computed the average difference between the past prediction and the future prediction by averaging the result of each remaining idle period probability comparison for each remaining idle period probability tested. Figure 4 shows this result for two successive months in part (a), and for the first 2.5 months and the last 2.5 months of our traces in part (b).

The comparison between traces from successive months shows that the average difference in predictions is



(a)



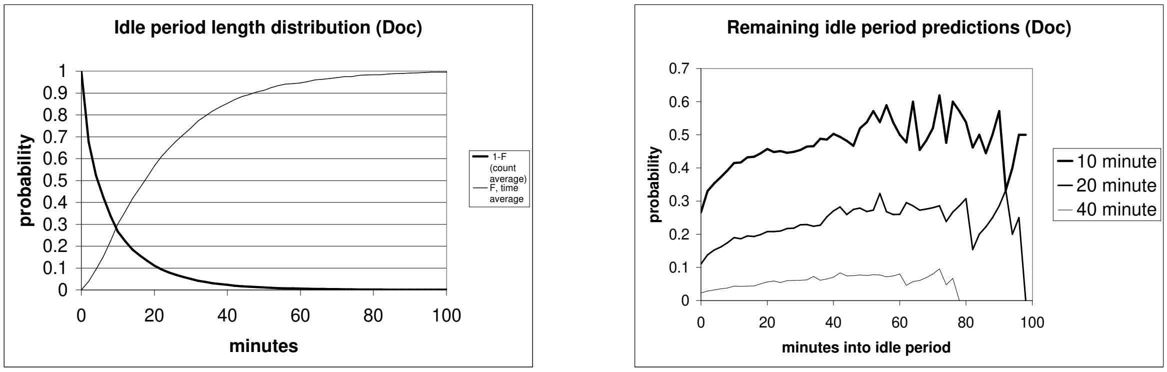
(b)

Figure 2: Idle period length distribution and remaining idle period prediction for Wilma.

less than .06 for more than two thirds of the workstations. When the comparison period spans the first and second half of the 5 month observation period, the differences in predictions are much larger, but almost two thirds of the workstations still have a prediction difference of .06 or less. Figure 5 shows the past and future remaining idle period probability prediction curves for two workstations with a moderately large average difference. These charts show the probability of finding a ten minute idle period for the past and future traces using the successive month partition. Part (a) shows the predictions for a workstation with an average difference of .061, and part (b) shows the predictions for a workstation with an average difference of .077. In both cases, the predictions are reasonably close, and provide reasonable estimates for finding long remaining idle periods. Furthermore, two thirds of the predictions are significantly better matches.

8.2 Workstations Show Different Behavior

It would be desirable to pool information about the idle period distributions from a collection of workstations. A statistically significant CDF can be constructed faster, and less space would be used to store the CDF. Unfortunately, doing so would give inaccurate results. We compared the remaining idle period predictions of workstations Ash, Doc, and Wilma against the other workstations. We generated average difference values for remaining idle period lengths of between 4 and 40 minutes, and averaged them. Figure 6 shows the results. Part (a) of the figure shows the sorted average difference values. This chart shows that each of the workstations Ash, Wilma, and Doc is significantly different (on average the difference is larger than .08) than two thirds of the other workstations. In part (b), we do not sort the average difference values, so that each value on the X axis refers to the same workstation in all three curves. Wilma and Ash have somewhat



(a) (b)

Figure 3: Idle period length distribution and remaining idle period prediction for Doc.

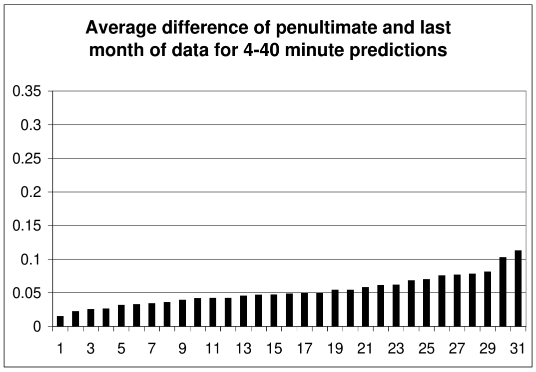
correlated average difference values, but this effect is weak.

8.3 Using the Current Time of Day

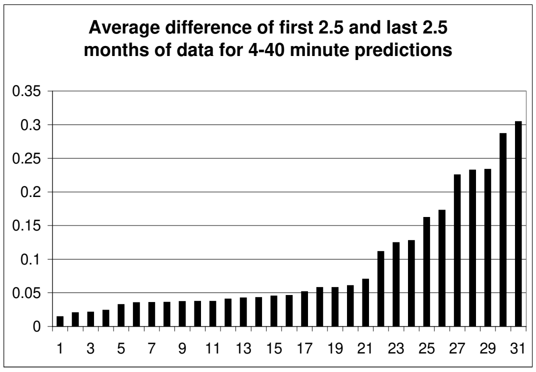
One would expect that knowing the current time of day can help in determining if the the current idle period is long. For example, one would expect long idle periods at night but short idle periods during working hours. We can incorporate time-of-day information into our remaining idle period length predictor by *partitioning* the traces based on the starting time of the idle period. We define a number of *time zones* (in our experiments, four) that partition time in a 24 hour clock. We partition the idle period trace by the time zone of the start of the idle period, and collect an idle period length distribution for each partition. If the time of day gives us information about the length of remaining idle periods, then we would expect that our method for comparing predictions gives large average difference values for the time zones.

There is a small technical problem with partitioning idle periods based on the time of day. Namely, a long idle period will cross several time zones. Since we allocate the idle period to the time zone in which it starts, we need to do the same when predicting the the probability of finding an idle period of length x. If we sample a workstation and find that it has been idle for i minutes and the idle period started in time zone k, then the probability that the idle period will last another x minutes, $P_k(i;x)$, is $(1-F_k(i))/(1-F_k(i+x))$, where F_k is the CDF of the idle period length distribution for time zone k.

To determine whether partitioning the idle period length CDF based on time zones is significant, we ran the following experiment. We partitioned the time of day into night (10:00 PM to 7:00 AM), morning (7:00 AM to 12:00 PM), noon (12:00 PM to 5:00 PM), and afternoon (5:00 PM to 10:00 PM). We compared



(a) Successive month



(b) First half vs. second half.

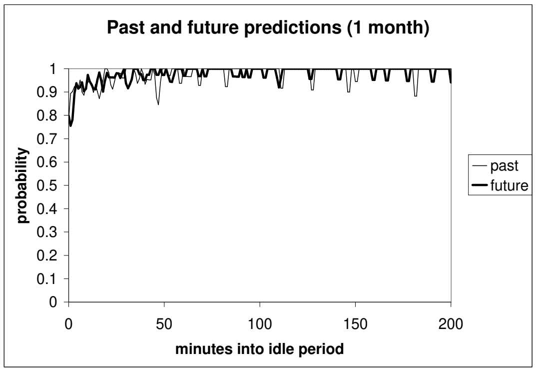
Figure 4: Average difference in past and future remaining idle period predictions.

the remaining idle period predictions against the non-partitioned remaining idle period prediction for 4 through 40 remaining idle minutes, and took the average. In Figure 7, we plot the maximum of the average differences of the comparison of the four time zones against the unpartitioned prediction. For almost two thirds of the workstations, there is a significantly large difference (average difference of .08 or larger) between the unpartitioned prediction and one of the partitioned predictions. We conclude that partitioning the idle period trace based on time of day can frequently improve the remaining idle period predictor.

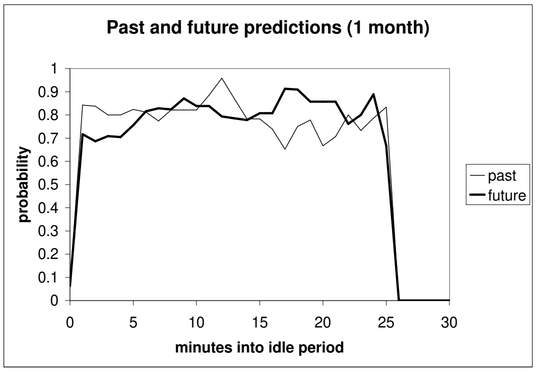
To provide an intuition on the nature of the differences that time zone partitioning provides, we present two remaining idle period length predictions in Figure 8. For both of these charts, we looked for ten minute remaining idle periods. In part (a) we show the predictions for each time zone, and the unpartitioned prediction for the workstation with the smallest average difference. Although here are significant differences between the unpartitioned and the zone 3 prediction towards the tail, our method of computing the average difference stops when the CDF of the idle period length reaches 99.5%. Hence, only the head is counted and the differences between predictions are not large in this region. Part (b) shows the predictions for the workstation with one of the largest average differences. Two of the time zones predict significantly fewer ten minute idle periods than the unpartitioned case does, and the other two predict more ten minute idle periods.

8.4 Using the Length of the Previous Idle Period

Another way to try to improve predictions of the remaining length of the current idle period is to use the length of the previous idle period. Previous researchers [3] found the lengths of successive idle periods to be



(a) Average difference of .061



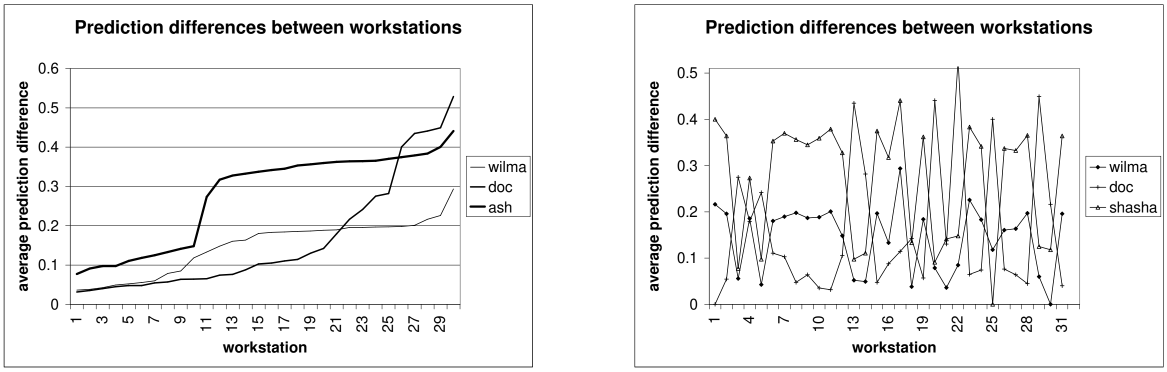
(b) Average difference of .077

Figure 5: Past and future predictions of 10 minute remaining idle periods for workstations with a large difference between past and future RIPPPs.

correlated. From this one can hypothesize that if the previous idle period was long, the current idle period will be long.

To test the hypothesis that the length of the previous idle period can help in predicting the length of the remaining idle period, we ran the following experiment. We partitioned the trace of idle periods from a workstation into three partitions, based on whether the previous idle period was short (less than 10 minutes), medium (10 to 30 minutes) or long (30 minutes or greater). We computed the difference between the partitioned remaining idle period predictions and the unpartitioned predictions for 4 to 40 minutes and averaged the results. In Figure 9, we show the difference in average prediction for the partition with the maximum difference from the average, for each workstation in sorted order. About one third of the workstations show a significant difference (.08 or larger) between the unpartitioned predictions and one of the partitioned predictions. We conclude that the length of the previous idle period provides some information on the probability that the current remaining idle period is long.

To show the differences in predictions that the length of the previous idle period provides, we chose two workstations and present their idle period distributions and 10 minute remaining idle period predictions. Figure 10, shows the idle period distribution (part a) and the 10 minute remaining idle period prediction (part b) for a workstation with a low average difference in prediction, and in Figure 11 we do the same for a workstation with a high average difference. These two charts show a surprising result. While the average length of an idle period that follows a long idle period is larger than average, one is not significantly more likely to find long remaining idle periods. In Figure 11 part (b), the probability of finding a long idle period



(a) Sorted by difference (b) Sorted by workstation name.

Figure 6: Differences between workstations.

is lower than average for most sampling points. This paradox is caused by fact that while short idle periods are less likely to follow long idle periods, the remainder of the idle period length CDF is not significantly different.

9 Implementation

In this section, we briefly describe our statistics gatherer and our idle period length prediction C library. These can be ftp'ed from our anonymous ftp site.

The data gathering daemon is written in Perl. As mentioned in section 4 it gathers load, keyboard, console, mouse, and remote logon statistics every minute. We have combined it with our conversion script to produce a data file of idle period lengths and/or busy period lengths. It generates a file called $hostname.tracelog.

The prediction library is written in C and provides functionality for doing both remaining idle (busy) length probability prediction for the three partitioning techniques described in the paper. Functions for computing the expected remaining idle (busy) length probability for each of the techniques are also provided.

10 Conclusion and Future Work

In this paper, we present a method for finding long idle periods in a network of workstations. In addition to finding a workstation that is likely to remain idle for a sufficiently long time, a RIPPP (Remaining Idle Period Probability Predictor) returns the probability that the idle period will be at least the desired length.

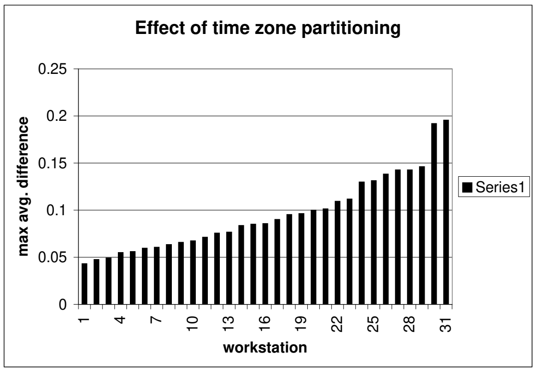


Figure 7: Average difference in RIPPPs using time zone partitioning (maximum difference shown).

This information is necessary to make optimization decisions about task placement, task migration, and recovery method actions.

We found that basing the RIPPP on time of day produced the best results. For all our RIPPP methods we showed that for a significant number of the workstations in our study, the past idle period length distributions did predict the future remaining idle period probabilities.

In the future we intend to base a non-dedicated network of workstations scheduler on our predictor. In order to do this, the following issues must be addressed: (1) the tails of the RIPPPs become inaccurate because of the lack of samples, (2) when gathering statistics in a real system, care must be taken to differentiate between the load generated by workstation owners and the harvested cycles, and (3) the idle period length distributions need to be automatically refreshed. These are technical, not fundamental difficulties. We believe that a scheduler based on these techniques will provide significant benefits over naive scheduling approaches.

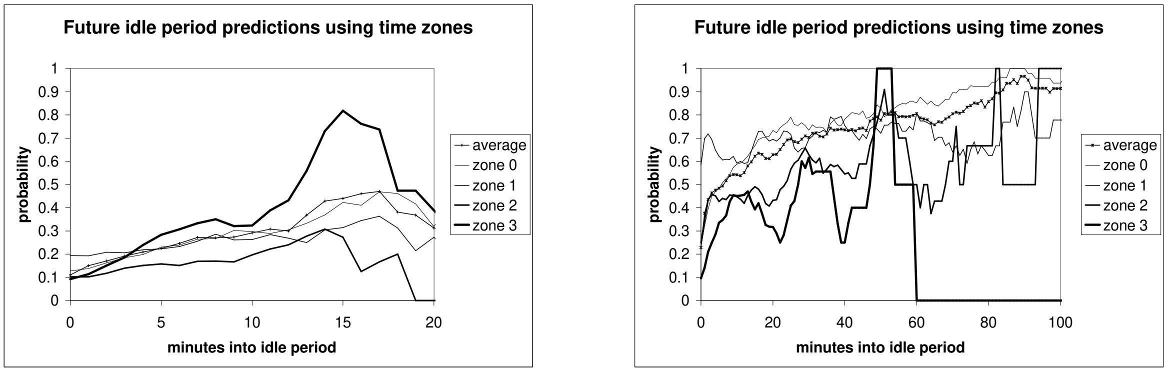
References

[1] R. Golding, P. Bosch, C. Stalin, T. Sullivan, and J. Wilkes Idleness is not sloth. USENIX Winter Conference, 1995. pg. 201-212.

[2] A. Acharya, G. Edjlali, J. Saltz. The utility of exploiting idle workstations for parallel computation. In ACM SIGMETRICS 1997. pg. 225-235.

[3] M. Mutka and M. Livny. Profiling workstations' available capacity for remote execution. In Performance'87. pg. 529-544.

[4] M. Samadani and E. Kaltofen. Prediction based task scheduling in distributed computing. In ACM Symposium on the Principles of Distributed Computing, 1995. pg. 261.



(a) Low difference in prediction

(b) high difference in prediction

Figure 8: Past and future predictions of 10 minute remaining idle periods for workstations with low and high differences in predictions between time zones.

[5] M. Harchol-Balter and A. Downey. Exploiting process lifetime distributions for dynamic load balancing. In ACM SIGMETRICS 1996. pg. 13-24.

[6] L. Kleinrock. Queueing Systems Volume 1: Theory. Wiley Interscience, 1975

[7] R. Agrawal and A. Ezzat. Location independent remote execution in NEST. *IEEE Transactions on Software Engineering*, 1987.

[8] R. Arpaci, A. Dusseau, A. Vahdat, L. Liu, T. Anderson, and D. Patterson. The Interaction of Parallel and Sequential Workloads on a Network of Workstations. In *Proc. of SIGMETRICS*, 1995.

[9] N. Carriero, D. Gelernter, D. Kaminsky, and J. Westbrook. Adaptive Parallelism with Piranha. Technical Report 954, Yale University Department of Computer Science, 1993.

[10] E. Chung, Y. Huang, and S. Yajnik. Checkpointing in CosMic: A User–level Process Migration Environment. In *Proc. of Pacific Rim Symp. on Fault-Tolerant Computing*, 1997.

[11] F. Douglis and J. Ousterhout. Transparent Process Migration: Design Alternatives and the Sprite Implementation. *Software – Practice and Experience*, 1991.

[12] D. Duke, T. Green, and J.Pasko. Research toward a heterogeneous networked computer cluster: The Distributed Queuing System Version 3.0. Supercomputer Computations Research Institute, Florida State University, 1994.

[13] M. Litzkow. UNIX: Turning Idle Workstations into Cycle Servers. In *Proc. Summer 1987 USENIX Conference*, 1987.

[14] D. Epema, M. Livny, R. Dantzig, X. Evers, and J. Pruyne. A Worldwide Flock of Condors : Load Sharing among Workstation Clusters. Journal on Future Generations of Computer Systems Volume 12, 1996

[15] M. Mutka and M. Livny. The Available Capacity of a Privately Owned Workstation Environment. *Performance Evaluation*, 1991.

[16] D. Nichols. Using Idle Workstations in a Shared Computing Environment. In *Proc. of SOSP*, 1987.

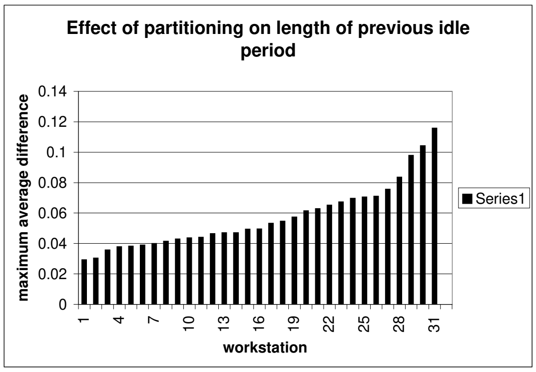


Figure 9: Average difference in remaining idle period predictions using previous idle period partitioning (maximum difference shown).

[17] J. Pruyne and M. Livny. Parallel Processing on Dynamic Resources with CARMI. In *Job Scheduling Strategies for Parallel Processing–IPPS'95 Workshop Proceedings*, 1995.

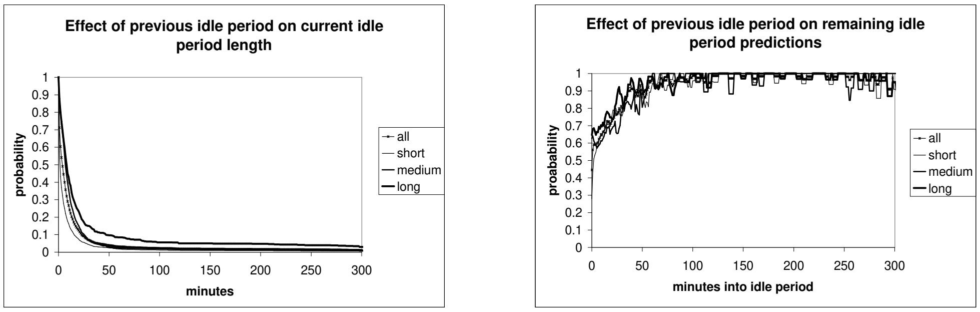
[18] J. Baldeschwieler, R. Blumofe, E. Brewer ATLAS: An Infrastructure for global computing. In *Proceedings of the seventh ACM SIGOPS European Workshop: Systems Support for Worldwide Applications*, 1996.

[19] R. Blumofe, C. Joerg, B. Kuszmaul, C. Leiserson, K. Randall, Y. Zhou. Cilk: An efficient multithreaded runtime system. In *Proceedings of the Fifth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 1995.

[20] T. Anderson, D. Culler, D. Paterson, et. al., A Case for Networks of Workstations (NOW). Computer Science Division, EECS Department, University of California, Berkeley, 1994.

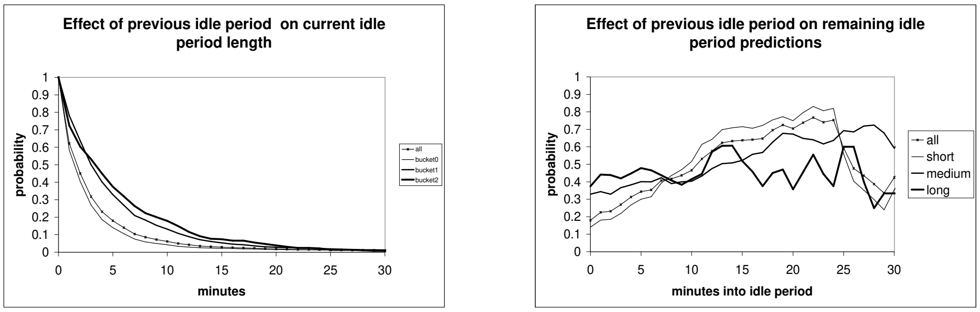
[21] D. Nichols. Using idle workstations in a shared computing environment. *ACM Operating Systems Review*, 21(5), 1987.

[22] M. Theimer and K. Lantz. Finding Idle Machines in a Workstation-based Distributed System. *IEEE Transactions on Software Engineering*, 1989.



(a) Low difference in prediction (b) high difference in prediction

Figure 10: Past and future predictions of 10 minute remaining idle periods for workstations with small difference in predictions for different previous idle period lengths.



(a) Low difference in prediction (b) high difference in prediction

Figure 11: Past and future predictions of 10 minute remaining idle periods for workstations with small difference in predictions for different previous idle period lengths.