Characterizing and Predicting Resource Demand by Periodicity Mining

Artur Andrzejak[*]
Mehmet Ceyran[†]

Running Head: "Characterizing and Predicting Resource Demand"

4th May 2005

[*]Zuse Institute Berlin (ZIB), Takustraße 7, 14195 Berlin-Dahlem, Germany, andrzejak@zib.de
[†]Zuse Institute Berlin (ZIB), Takustraße 7, 14195 Berlin-Dahlem, Germany, ceyran@zib.de

Abstract

We present algorithms for characterizing the demand behaviour of applications and predicting demand by mining periodicities in historical data. Our algorithms are change-adaptive, automatically adjusting to new regularities in demand patterns while maintaining low algorithm running time. They are intended for applications in scientific computing clusters, enterprise data centers, and Grid and Utility environments that exhibit periodical behaviour and may benefit significantly from automation. A case study incorporating data from an enterprise data center is used to evaluate the effectiveness of our technique.

Keywords: prediction, pattern mining, resource management, self-management

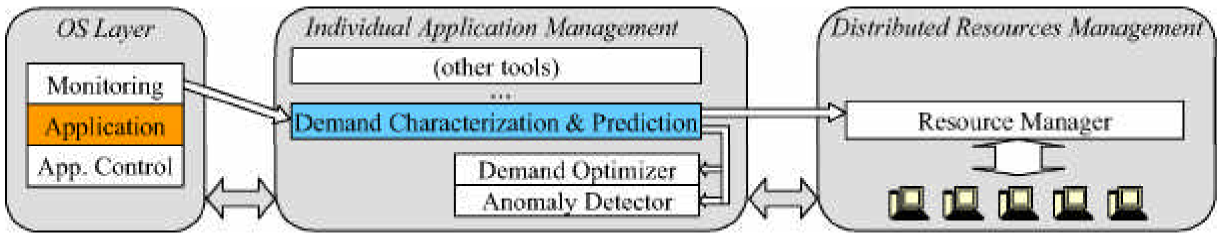


Figure 1: Role of demand characterization and prediction in a framework for automated system management

1 Introduction

Characterizing and predicting the demand of individual servers, clusters, or enterprise data centers paves the way for automated management and scheduling of computing resources. On the level of individual applications, demand tuning and anomaly detection are greatly facilitated by increased knowledge of demand. Proactive scheduling and automated resource consolidation processes can also benefit enormously from demand prediction. Such processes are becoming increasingly important in reducing operating costs and simplifying the management of large resource pools such as enterprise or utility data centers.

Periodicity-based characterization and prediction techniques work only if demand characteristics observed in the past are likely to prevail in the future. The nature of the applications in a system must be considered as a deciding factor in this assumption. Applications employed in business or scientific domains are especially likely to exhibit repetitive demand behavior. Examples of such resource consumers (i.e. applications or services) are business always-on applications or scientific computing applications. Our work and the results of this paper focus on these types of applications and scenarios which involve them.

The contributions of this paper are as follows. We develop a set of methods for capturing periodic regularities in the behavior of resource consumers. The major advantages of our approach are change-adaptivity, low running time compared to known methods, and compact representation of the mining results. Furthermore, we propose a computationally efficient technique for predicting future demand. Under the assumption that demand behaviour does not change significantly in the future, this technique allows us to accurately predict short- and long-term demand. We evaluate our approach with a case study using synthetic traces and traces collected from data centers.

1.1 Applications for Self-Management of Systems

We envision demand characterization and prediction processes as part of a framework for automated management of applications and resources. While the foci of this paper are the specific methods of characterization and predication and not the architecture of such a framework, it is worth considering the benefits of a demand characterization and prediction tool in the context of this framework.

Managing individual applications. The proposed periodicity mining approach provides a compact characterization of demand with low computational cost. As illustrated in Figure 1, periodicity mining results can be used by external components to tune and manage individual applications. Two example components are shown in the figure: a demand optimizing module and an anomaly detector.

A demand optimizing module usually has policy-driven goals such as preventing server overload, keeping the demand curve constant or general reduction of the cumulative demand. Data on demand characteristics greatly enhances such a tool by allowing it to take preventive action, like preprocessing data before expected demand surges. This information might also help to prevent demand oscillations caused by interactions between several applications.

An anomaly detector compares demand characteristics with actual behaviour for discovering unusual situations, such as unexpected demand peaks or prolonged idle periods. Demand may also be considered with other data in the automated detection of application or system component breakdowns, or e.g. Denial of Service Attacks.

Managing distributed resources. Demand prediction capabilities enable *proactive* resource management in a distributed environment. In contrast to classical reactive resource management, proactive management schedules the tasks or applications to resources in anticipation of the expected demand surges or idle times. As a result, the number of hot-spots is reduced and the degree of resource utilization increases.

In addition short-term scheduling, a tool for demand characterization and prediction might also be used for automated "matching" of applications with complementary demand behaviour for server consolidation - several consumers can use the same resource as long as the respective phases of high and low demand balance each other out. A study in [11] (using off-line scheduling) confirms the high potential of this approach for Utility Computing and data center environments.

Finally, data mined from historical traces can be used to automatically classify applications into those with predictable behavior and those with less predictable demand characteristics. The potential savings of resource sharing is directly proportional to the predictability of the consumers using the resources. An application displaying less predictable behavior might require a dedicated resource to prevent overload or to satisfy a Service Level Agreement (SLA) [12], even if the consumer's average demand is low. Thus, the predictability level can consulted in automated placement on dedicated resources, and might also be considered in the pricing of a SLA.

1.2 Approach Details

Change-adaptivity. A key feature of our methods is their *change-adaptivity.* A mining method with change-adaptivity favors recent demand behavior characteristics over older ones. When making predictions based on historical data, the following trade-off must be considered: a large amount of historical (training) data enables more accurate prediction, but the increased size of this data slows adaptability to new demand behavior, since older regularities may "overwhelm" new (and thus more appropriate) patterns. We attempt to solve this problem by weighting historical data according to "age". Weighting the data in this fashion isolates stationary behavioral patterns as more data becomes available, though new characteristics still

Range top	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000
Probability	0.08	0.42	0.17	0.12	0.12	0.04	0.00	0.04

Table 1: A resource demand prediction represented by a probabilistic mass function (pmf)

dominate older ones, even if the former appear in only a fraction of the data.

Periodicity mining. To capture periodic regularities, we divide the signal amplitude into ranges, which we call *levels*. For each level we record the events of a signal amplitude belonging to this level. This sequence of events (differing only in their timestamps) is subject to several mining steps. The most important of these steps focuses on mining periodicities and the corresponding phase shifts. For mining periodicities, we extend an algorithm by MA and HELLERSTEIN [9] to efficiently find statistically significant periodicities yet ensure change-adaptivity.

Predicting demand. The second part of this paper focuses on the prediction of application demand. Given a time instant in the future (relative to the historical data), we can compute a *probability mass function* (*pmf*) [1] of the demand. This type of forecast approximates the probability distribution of the demand by describing the probability that it is within r, for each of a finite number of demand value intervals r. Table 1 shows an example forecasted pmf for the processing demand of an application in a business data center. The demand is normalized to the interval $[0.0, 1.0]$. The advantages of this approach are low running time and the fact that the predictions can reach further into the future without compromising accuracy (assuming the behavior profile of a consumer does not change). These are substantial advantages compared to currently employed methods such as ARIMA or ARFIMA modeling discussed in Section 2.

The body of the paper is organized as follows. In Section 2 we discuss related work. Section 3 is devoted to mining characteristics of repetitive demand behavior. We describe a method of using such statistics to generate predicting pmf's in Section 4. A case study is discussed in Section 5. Finally, we conclude with a summary in Section 6.

2 Related Work

Periodicity Mining. One traditional approach to detecting periodicities is based on an analysis of the signal frequency spectrum obtained by Fast Fourier Transforms (FFT). The unmodified analysis has the disadvantage that frequency changes over time cannot be detected, a problem that can be remedied by applying Short-term Fourier Transform and wavelet analysis [14]. The downside of the Short-term FT is the difficulty to choose the size of a "sliding window" where FFTs are applied.

In the field of data mining there has been a considerable interest in mining periodic or partially periodic patterns [6], [9]. Targets of these studies include database mining or web log analysis. An interesting extension of this work is mining of hierarchical patterns [15]. This approach could also enrich our model for describing periodicity. However, hierarchical pattern mining might also lead to spurious results since more complex patterns are less likely to reoccur in the future.

Demand Prediction. Work on resource demand prediction includes calendar methods [7], also used in [12]. Here a fixed period (e.g. 1 day) is assumed, and a probabilistic profile for each chunk of the period (e.g. for 1pm-2pm, 2pm-3pm etc.) is computed. One drawback of these methods is a lack of flexibility when several periodicities are present. Also, the dominant periodicity is determined in an arbitrary way.

More advanced approaches to demand prediction borrow from econometrics and use time series models based on auto-regression and moving averages such as ARIMA and ARFIMA [5], [13]. The most comprehensive study of this technique was conducted by DINDA [4], who also developed an extensible toolkit for resource prediction in distributed systems [3] that implements the above-mentioned methods as well as wavelet-based analysis tools [14]. These approaches yield good short-time (around 30 samples) prediction, but fail to capture longer-term regularities. This is due to the fact that the error interval for these methods grows relatively quickly the further a prediction goes beyond the end of the data. Moreover, to capture a periodicity of a seasonal behavior of size τ, the auto-regression depth must be at least τ. Two problems arise here: first, one must know the expected period length in advance. Second, the computational time increases drastically with the depth of the auto-regression. In contrast, our approach has the advantage that long-term predictions do not necessarily have larger error than shorter-term ones. Furthermore, the observed running time is linear in the number of input samples with a low constant, and does not depend on the size of mined periodicities.

Further studies of demand prediction have concentrated on the power generation sector [2] and network traffic management [10].

3 Change-Adaptive Periodicity Mining

During the preprocessing phase, the input signal is split into event sequences corresponding to subranges of its amplitude (level). Each sequence is then converted into multiple series of contiguous events (segments). The goal of the subsequent mining phase is to identify (sub)sequences of segments with statistically significant periodicities for *each level.* Furthermore, we want to extract enough information about these periodic components to be able to extrapolate their behavior beyond the known data.

Several mining steps are necessary to achieve this goal. First, a change-adaptive algorithm mines segment *periodicities* at each level. The resulting periodicities serve as the basis for deriving further characteristics of a sequence. The efficient and change-adaptive methods presented in this paper are then used to compute the *phase shift* of a periodic segment sequence (in relation to a sequence with "peak" at $0s$), and the *mean* and *variance* of the segment widths.

3.1 Input Signal and Preprocessing

Our input data is a single *time series*, defined as a sequence of pairs (timestamp, value) ordered by timestamps. We assume that the timestamps are equidistant and write D for the time difference of two consecutive timestamps. This assumption is not essential to our algorithm, but simplifies the implementation by making it robust in the presence of occasional outliers. The value of a time series is assumed to be a function of the demand of some resource sampled

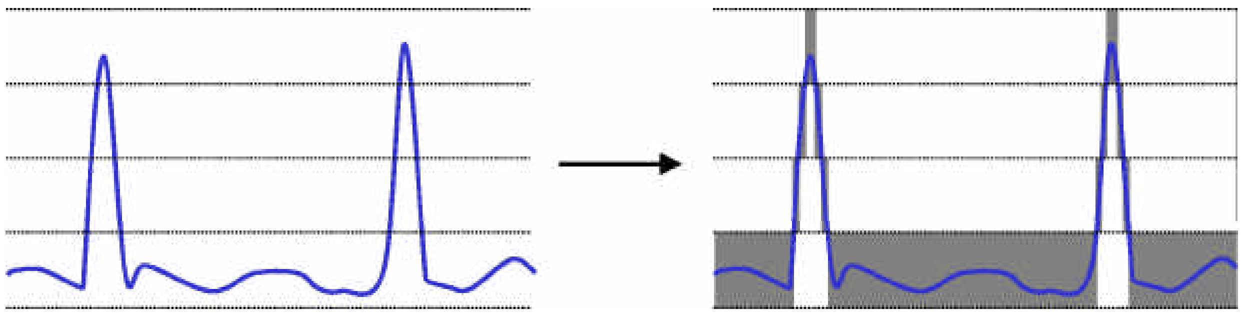


Figure 2: Input signal with levels and its conversion to four sequences of segments (x-axis: time, y-axis: signal value)

each D timestamps (D is called *granularity*). Examples of this function are CPU utilization averaged over a given time period D, or the demand of the resource at the moment of sampling. In general, we refer to the input time series as a *signal*, to emphasize the fact that the methods presented here are applicable to any kind of data. The only restriction is that the values in the time series lie in a specific, limited range.

In the preprocessing phase, we subdivide this range of values into L disjoint intervals called *levels*. While these levels have equal size in the current implementation, ranges with different sizes could be used as well. A value v of the pair (timestamp, value) is mapped to a level such that v is in the interval of this level. For each level k we obtain the sequence of the timestamps corresponding to the occurrences of values within k (again ordered by timestamps). Thus we transform our input signal into L sequences of (equal) *atomic events*. The choice of the number of levels depends on the desired accuracy and the nature of the input data. For example, we might choose 64 levels for the computational demand of a 16 CPU machine, and 8 levels for a dual processor machine.

The sequence of atomic events in a level is further decomposed into *segments*. Segments are contiguous sequences of timestamps such that the difference of adjacent timestamps is not larger than a predefined number $k\,D$, with k some small integer greater than 0. The goal is to cluster together consecutive events while tolerating $k-1$ "drop-outs" within a segment. A segment is uniquely specified by its *width* (the difference end-begin) and its *center* ((begin + end)/2). Figure 2 shows an input signal, four levels, and the obtained sequences of segments (each represented by a grey box).

For the following sections we require formal definitions of the concepts described above. Consider a sequence of segment centers $e_0, e_1, \ldots$ such that the timestamp $e_i.t$ of e_i is $i\tau + s$, where τ and s are non-negative numbers with $s < \tau$. For convenience, we refer to a segment center e also as an event. We call τ the *periodicity* of the sequence, and s its *phase shift*. For a given τ and s, we call a timestamp $i\tau + s$ a *hit point*.

3.2 Periodicity Mining

The following section presents a method for discovering statistically significant periodicities of segments for a given level. The method is based on the algorithm presented in [9]. We extend it by adding change-adaptability. Our input is the sequence $e_1, e_2, \ldots$, of $N+1$ segment centers of a single level with corresponding timestamps $e_1.t, e_2.t, \ldots$.

Algorithm by Ma and Hellerstein. We start by describing the original algorithm published in **[9]**. It consists of three processing phases:

1. First, the set of potential "raw" periodicities is computed. For each pair of consecutive events e_i and e_{i-1}, this is the difference $\tau = e_i.t - e_{i-1}.t$ (inter-arrival time). Then a counter C_τ for this particular τ is created and initialized to 1 (if C_τ did not exist), or incremented by 1 (if C_τ did exist). The output of this phase is the set of observed τ's (potential periodicities) together with their respective counters C_τ.

2. Second, the counters C_τ are adjusted to incorporate the *tolerance* δ for periodicities. This parameter is given as input. In the beginning, all $C'_\tau s$ (together with the corresponding $\tau' s$) are in the candidate set, and the result set is empty. We repeat the following greedy procedure until the candidate set becomes empty. For every τ in the candidate set all counters $C_{\tau'}$ for τ' within an interval $[\tau - \delta, \tau + \delta]$ are added together to a new counter $C_{\tau,\delta}$. Then the largest $C_{\tau,\delta}$ (and the corresponding τ) are moved to the result set, and all $C'_\tau s$ and their $\tau' s$ which contributed to $C_{\tau,\delta}$ are removed from the candidate set.

3. Finally, the pairs (τ, C_τ) in the result set of phase 2 are checked for statistical significance. We discard τ if $C_\tau \leq C'_\tau$ for a threshold C'_τ whose computation is stated below. The remaining values of τ constitute the set of the statistically significant periodicities, the output of the algorithm in **[9].**

C'_τ is computed as follows. Let S be a sequence of events with random inter-arrival time (time between two consecutive events), such that the mean inter-arrival time in S is the same as the mean inter-arrival time in the mined sequence of events. Let p_τ be the probability that the inter-arrival time between two events in S lies in the interval $[\tau - \delta, \tau + \delta]$. It is obvious that Np_τ is the expected number of instances of inter-arrival times in the interval $[\tau - \delta, \tau + \delta]$ in S, and the variance of this number is $Np_\tau(1 - p_\tau)$. Recall that $N+1$ is the number of segment centers in the whole sequence.

Intuitively, the larger C_τ is in relation to Np_τ, the less likely that the periodicity τ is due to random phenomena only. We can turn this difference into statistical significance by using chi-square statistics [1]:

$$\chi^2_\tau = \frac{(C_\tau - Np_\tau)^2}{Np_\tau\,(1 - p_\tau)}. \tag{1}$$

The value of χ^2_τ is the normalized deviation from the distribution of the sequence S. Depending on the degrees of freedom (one in our case), the value of χ^2_τ determines the probability p_{err} that the computed value of C_τ or larger occurs in a sequence S (i.e. the value of C_τ is due to randomness only). For example, for $p_{err} = 0.05$ we have $\chi^2_\tau = 3.841$, and for $p_{err} = 0.005$

we have $\chi^2_\tau = 7.879$. Because p_{err} (the probability that we accept a periodicity that is due to random phenomena only) is inversely proportional to χ^2_τ, the corresponding C_τ must be higher in order to attain an acceptable τ. By choosing p_{err}, substituting the corresponding value for χ^2_τ and solving (1) for C_τ, we obtain the desired threshold $C'_\tau = C_\tau$.

We still need to compute the probability p_τ. With an increasing number of events the distribution of inter-arrival times in S approaches the exponential distribution. With $\lambda = N/T$ being the mean inter-arrival time of S (where T is the time span of this sequence) and assuming that δ/τ is small, it holds $p_\tau = \exp(-\lambda(\tau - \delta)) - \exp(-\lambda(\tau + \delta))$.

Incorporating change-adaptivity. Instead of allowing each occurrence of the inter-arrival time $\tau = e_i.t - e_{i-1}.t$ to make the same contribution to C_τ, we weight each occurrence of the inter-arrival time τ. For an event e_i with the difference $e_i.t - e_{i-1}.t = \tau$ let

$$w_i = w(e_i) = 2^{-a/T_h} \tag{2}$$

be the *weight* of e_i. Here a is the difference between the last timestamp in the input time series and $e_i.t$ (the "age" of e_i), and T_h is a parameter specified in the input. Note that if the age of e_i is kT_h, then $w(e_i) = 2^{-k}$, and so we call T_h the *half-life* of the weight. For a given τ, the role of C_τ is then replaced by the sum of weights

$$W_\tau = \sum_{e_i.t - e_{i-1}.t \in [\tau-\delta, \tau+\delta]} w(e_i), \tag{3}$$

i.e. this value is computed in step 1 for each observed τ.

This approach can be used in steps 1 and 2 of the above algorithm without essential changes. However, in step 3 we need a new schema to compute the threshold C'_τ, now called W'_τ. The idea is to derive W'_τ from the mean and variance of weights in the sequence S of events with random inter-arrival times (defined above).

Let $e = e_i$ be an event in S. We define X_{τ,e_i} as a random variable (r.v.) whose value is $w(e_i)$ if $e_i.t - e_{i-1}.t \in [\tau - \delta, \tau + \delta]$, and 0 otherwise. With the probability p_τ as above, the expectation $E[X_{\tau,e_i}]$ is $w(e_i)p_\tau$, and its variance $V[X_{\tau,e_i}]$ is $w^2(e_i)p_\tau$, as X_{τ,e_i} has a binomial distribution. Let X_τ be the sum of random variables X_{τ,e_i} for the whole sequence S. It is not hard to see that for a specific τ the value of W_τ in (3) averaged over many random sequences S approaches $E[X_\tau]$. The value of $E[X_\tau]$ is $p_\tau \sum_{i=1}^N w(e_i)$ since the expectation is additive. In order to apply chi-square statistics, we also need to know the variance $V[X_\tau]$. By observing that the events in S are independent, we obtain

$$V[X_\tau] = \sum_{i=1}^N V[X_{\tau,e_i}] = p_\tau(1 - p_\tau) \sum_{i=1}^N w^2(e_i).$$

Now we can use the relation

$$\chi^2_\tau = \frac{(W_\tau - E[X_\tau])^2}{V[X_\tau]}. \tag{4}$$

We obtain the threshold $W'_\tau = W_\tau$ by choosing the desired p_{err}, substituting the corresponding value for χ^2_τ and solving (4) for W_τ.

The problem of efficiently computing the sums $\sum_{i=1}^N w(e_i)$ for expectation and $\sum_{i=1}^N w^2(e_i)$ for variance of X_τ remains. We could generate a sequence S with (pseudo-)random inter-arrival times, and the same mean inter-arrival time as that in the mined sequence. However, we can also approximate the above sums by the corresponding sums computed over non-degenerate (in the sense of non almost constant) input sequences, for the following reasons. First, if the mean inter-arrival time is small compared to T_h (which is the case), then small deviations of the timestamp values of e_i produce only a negligible change in $w(e_i)$. Second, if the events in the input sequence are equally spread over its time span (as in the case of non-degenerate sequences), then the weights of the ith events will be similar in both an input sequence and a sequence S. Summarizing, the above sums can be substituted by corresponding sums over the input sequence without essential errors.

The change-adaptive algorithm described above might output several statistically significant periodicities for an event sequence associated with a level. For the purposes of the prediction algorithm from Section 4 only one of these periodicities is necessary. We select it by comparing the ratios W_τ/W'_τ, and taking the τ with the highest ratio as the dominant periodicity of the respective level.

3.3 Characterizing Segment Sequences

The periodic behavior of a sequence of events is not completely characterized by its periodicity. In order to "replay" a sequence of segments as defined in Section 3.1, we need to determine the phase shift of the sequence of the segment centers (as compared to a sequence with a segment center at $0 \bmod \tau$). Furthermore it is necessary to characterize the segment widths in a change-adaptive way, see Figure 3. We must also compute the probability that a hit point determined by the computed periodicity and phase shift actually occurs within a segment of the training sequence.

Computing phase shift. Recall that in an ideal sequence of segment centers with periodicity τ, the phase shift is the remainder of the division of any timestamp by τ. However, the real sequence might have several subsequences with periodicity τ but different phase shifts. In extending the concept of change-adaptivity, we define the *dominant phase shift* s as the phase shift of the subsequence with periodicity τ which has the largest sum of weights (defined by (2)) of its events.

We compute the dominant phase in a single iteration over the sequence. We use a dictionary Dic with keys being possible phase shifts. For each segment we first check whether the segment width allows the segment center e to be in a sequence of periodicity τ. If yes, we compute the remainder of $s' = e.t \bmod \tau$ and add the weight $w(e)$ to the entry under the key s' in Dic. When the iteration is complete, the value for key s' in Dic is the sum of weights of events for the subsequence with phase shift s'. The key s with the largest value can then be found in logarithmic time in the size of Dic if a tree data structure is used for Dic.

Characterizing segment widths. After the above computation, the sequence of segment centers is characterized by τ and s. The computation of the change-adaptive mean m_w and variance var_w of the segment widths is executed as follows. For each segment center we check whether its width allows the segment to be in a sequence of periodicity τ. If this is the case, we check whether the timestamp t of the segment center is within distance δ from a hit point $i\tau + s$. If this also holds and the hit point $i\tau + s$ has not been "used" before (this could happen if several segment centers are within δ of this hit point), the focus segment contributes to the

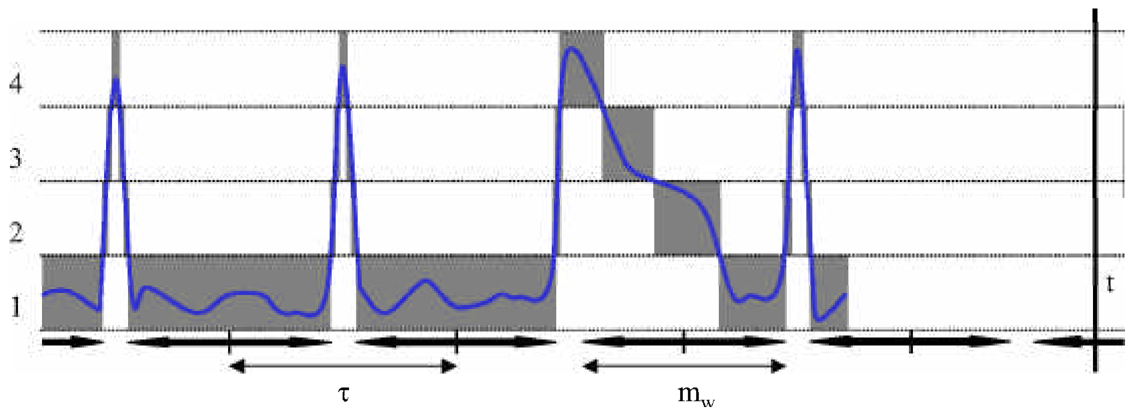


Figure 3: The periodicity characteristics captured for level 1 (x-axis: time, y-axis: signal value)

mean and the variance of the widths.

To ensure change-adaptability we use the following schema for computing mean and variance. We treat the width of a segment as a discrete random variable X. Assuming that each segment has a unique width, the per-event probability $p(e)$ of X attaining the value v (the width of a segment with center $e.t$) is defined as the quotient $w(e)$ divided by the sum of the weights of all segment centers that contributed to the computation (here $w(e)$ is again determined by (2)). Then the mean m_w is computed according to the standard formula for the expectation of a r.v. (i.e. $m_w = E[X]$), and var_w by the standard formula of a variance of a r.v., $var_w = V[X]$.

Computing hit probability. The values of τ, s, m_w and var_w give us a model of a periodic subsequence of segments contained in the input data. However, the input data might produce fewer segments than the model predicts, see Figure 3. To measure the accuracy of the model, we introduce the *hit probability* p_h, defined as follows. Let w_{max} be the sum of weights $w(e)$ defined by (2) over all segment centers e that occur in the "ideal" segment sequence described by the model. Let w_{act} be the sum of weights $w(e)$ over those segment centers e that fulfill two conditions: 1) $e.t$ is within distance δ from a hit point $i\tau + s$ prescribed by the model parameters, and 2) the segment corresponding to e has a width that allows this segment to be in a sequence of periodicity τ. Then the hit probability is the ratio

$$p_h = w_{act}/w_{max}.$$

The value of w_{act} is computed by looping over all segment centers and applying the same selection process as that used in computing width characteristics. To compute w_{max}, notice that the last segment prescribed by the model has an approximate weight of 1 (since its age is almost 0). Furthermore, we obtain the $(i-1)$th weight w_{i-1} by multiplying the ith weight w_i by $q = 2^{-\tau/T_h}$. If $z+1$ is the number of segments determined by the model, then the value of w_{max} is given by the geometric sum formula $w_{max} = \sum_{i=0}^{z} q^i = \frac{q^{z+1}-1}{q-1}$.

3.4 Running Time

The running time of the periodicity mining algorithm is dominated by step 2 (adjusting the periodicities according to the tolerance δ). A rough analysis shows that the worst-case running time is $O(N^3)$ (N being the number of segments in a level). However, in all practical cases computing a single tolerance-adjusted τ requires on average $2\delta/N$ steps, and the final number of adjusted τ's is small compared to N. Therefore, the observed running time was $O(N)$.

Mining the phase shift takes at most $N \log N$ steps, while computing width characteristics and the hit probability can be done in linear time.

4 Predicting Probabilistic Profiles

The computation of the probability mass function (pmf) of future demand discussed in this Section is based on the statistics derived in Section 3. The general idea is to determine for a given future instant t the level k that is most likely to have a segment covering t (as predicted by the periodicity statistics). We then return the pre-computed pmf characterizing the demand behavior within the segments predicted for k. In Figure 3 the instant t is covered by a segment of level 1, and so we would return the pmf computed over all segments determined by the periodicity model for level 1 as the result for t.

We will describe the latter pmf (called *hit pmf*) in more detail. Consider a level featuring a periodicity. For a given timestamp t we can easily compute whether t might be within a segment belonging to the captured periodic sequence (i.e. if t is in the interval $[i\tau + s - m_w, i\tau + s + m_w]$ for some integer i), or if this is unlikely. In a former case a [change-adaptive] hit pmf of the input signal computed over the intervals $[i\tau + s - m_w, i\tau + s + m_w]$ is likely to correctly describe the probability distribution of the demand at the instant specified by t (here i takes on values such that all intervals of the above form in the input data are covered). In Figure 3 the hit pmf for level 1 might be 0.7, 0.1, 0.1, 0.1 (probability for levels $1, \dots, 4$ respectively).

To resolve the question of which hit pmf is most likely to capture the probability distribution at instant t (if any at all), we introduce a *hit indicator* for each level. The hit indicator's value is 1.0 if t is within the interval $[i\tau + s - m_w, i\tau + s + m_w]$ for some integer i, and it decreases to 0.0 beyond the limits of this interval according to the Normal distribution. After calculating the hit indicators for all levels that feature periodicity, we choose the hit pmf of the level with the highest hit indicator value as the representative description of the demand profile for the instant t (ties are resolved by preferring levels with larger hit probability p_h). In Figure 3 the hit indicator for level 1 is obviously 1.0, since t is within the segment prescribed by the model. If the hit indicators of other levels were less than 1.0, the hit pmf of level 1 would constitute the prediction result.

There might often be no level with a hit indicator above 0.0. In this situation, the best bet is to use a [change-adaptive] pmf computed over the whole input data (called *histogram pmf*) as the pmf for the requested instant. If the largest hit indicator for t has a value v between 0.0 and 1.0, we use a linear combination of the appropriate hit pmf and the histogram pmf weighted by v.

4.1 Histogram Pmf and Hit Pmf

In the standard (non change-adaptive) version, the *histogram pmf* is a discrete random variable which takes as values the midpoints of the signal amplitude ranges corresponding to the levels (e.g. 0.125, 0.375, 0.625 and 0.875 for four levels subdividing $[0, 1]$). The probability corresponding to the level k is the number of sampled points (events) belonging to level k divided by the total number of the sampled points (over all levels). Note that we use the "raw" sampled points here (or pairs (timestamp, value) defined in Section 3.1) and not the segments.

To make this definition include change-adaptation, we simply substitute the count of a sampled point by its weight (defined by (2)). In other words, the probability corresponding to a level k is the sum of weights of the sampled points falling into level k divided by the sum of weights of all sampled points.

The *hit pmf* for a level k is computed in the following way. Given a periodicity model for level $k = 1, \ldots, L$ with parameters τ, s and m_w, we scan all sampled points and check whether the timestamp of such a point e falls into the interval $[i\tau + s - m_w, i\tau + s + m_w]$ for some integer i. If this is the case, we add the weight $w(e)$ to the sum $s_{k'}$ for the level k' to which e belongs. After the scan, the probability for a level k' is $s_{k'}$ divided by the sum of weights of all used points (i.e. divided by the sum $s_1 + \ldots + s_L$).

4.2 Hit Indicator and Resulting Pmf

The computation of the hit indicator for a level is relatively simple. If a level has no detected periodicity, then the returned value is 0.0. If a level has a periodicity we check whether the input t is in the interval $[i\tau + s - m_w, i\tau + s + m_w]$ for an integer i. If this is the case, the value of the hit indicator is 1. Otherwise, we find an integer i such that one of the numbers $i\tau + s - m_w, i\tau + s + m_w$, say b, is closest to t. The value of the hit indicator is then the normalized value of the probability density function of the Normal distribution with mean b and variance var_w. The normalization factor is $\sqrt{2\pi(var_w)}$ so that the output is 1.0 for $t = b$.

To obtain the final result, we find the level with the highest indicator value v. If $v = 1.0$, we return the hit pmf of the corresponding level, and if $v = 0.0$ then the histogram pmf is returned. Otherwise the result is a linear combination between a hit pmf (weighted by v) and the histogram pmf (weighted by $1 - v$).

4.3 Running Time

The execution of the algorithm is split into two steps, preprocessing and an on-line pmf computation for a requested time instant t. In the preprocessing step we first compute periodicity statistics for each level, as described in Section 3. The worst-case running time for this computation is $O(N^3)$ per level, where N is the number of segments in the respective levels. As noted above, the observed average time is $O(N)$. The number of segments is usually much less than the number of sampled points, and the levels have disjunct sets of segments, so in practice the complete periodicity mining requires linear time in the number of samples n. After computing periodicity statistics we calculate the histogram pmf (one per trace) and the hit pmf for each level featuring periodicity. Each pmf computation requires exactly n steps, so that this processing part requires at most $n(L + 1)$ steps. In real time this usually takes longer than the periodicity mining.

Given a request to compute a pmf for a time instant t, we need to compute the hit indicators for all levels featuring periodicity, and output the resulting pmf according to the schema

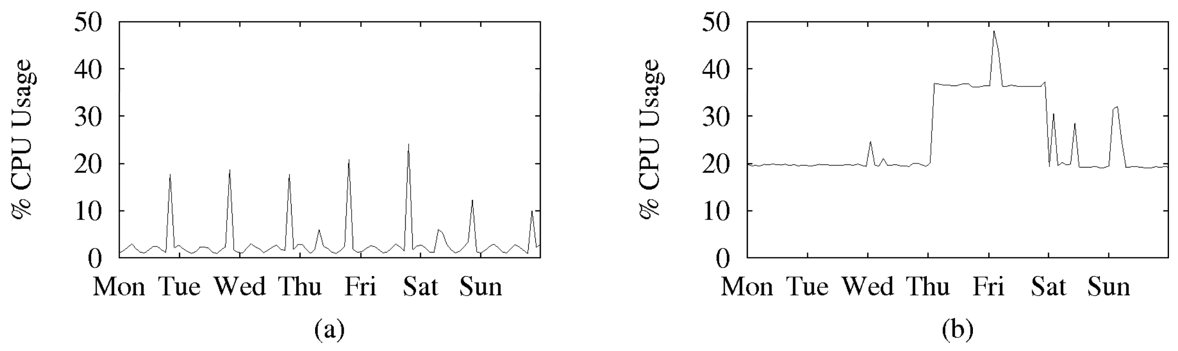


Figure 4: Weekly traces of data set B

described in the beginning of this section. This obviously takes only $O(L)$ time.

5 Evaluation

In this section we present the results of an evaluation of our algorithm using real and synthetic data sets. Each data set has been split into a training part (first 80% of the samples) and a test part. We have the periodicity mining tolerance δ depend on the candidate periodicity τ by introducing the *relative tolerance* δ_r such that $\delta = \tau\delta_r$. If not otherwise stated, this value was set to $\delta_r = 0.1$ and the value of p_{err} to 0.01 for the whole study.

5.1 Systems Under Study

In order to evaluate our approach we employ a set of synthetic data (Data Set A) and a set of real CPU utilization data (Data Set B).

Data Set A. The synthetic data set has been used for debugging purposes and for testing the change-adaptive features. We have incorporated two classes of signals. A *sin*-class signal has as the underlying function $f(x) = \text{abs}(\sin(x2\pi/\tau))$, where τ is the desired periodicity. The *pulse*-class signal is a rectangular signal of periodicity τ which alternates between 0.0 and 1.0 and has the desired pulse width w (i.e. in each period the fraction w of the period has amplitude 1 and the remainder has amplitude 0). The generated signal had timestamps from 0 to 2592000 seconds (30 days) and granularity $D = 300$.

Both classes of signals were tweaked to make them less regular. With probability p_n we added a random value from $[0, 1]$ to a sample and truncated the sum to 1. Furthermore, the first part of the signal (fraction h of the whole length) could have a different periodicity and pulse width (for the pulse-class) than the remaining part.

Data Set B. This set contains CPU utilization information from 41 servers in a data center. This information was collected using HP MeasureWare (Openview Performance Agent) between July 29th, 2001 and September 2nd, 2001 and it has been used in the study in [11]. We treat each server as an application (demand consumer). This approach is dictated by the fact that in most data centers resource sharing is uncommon and so each application has a dedicated server.

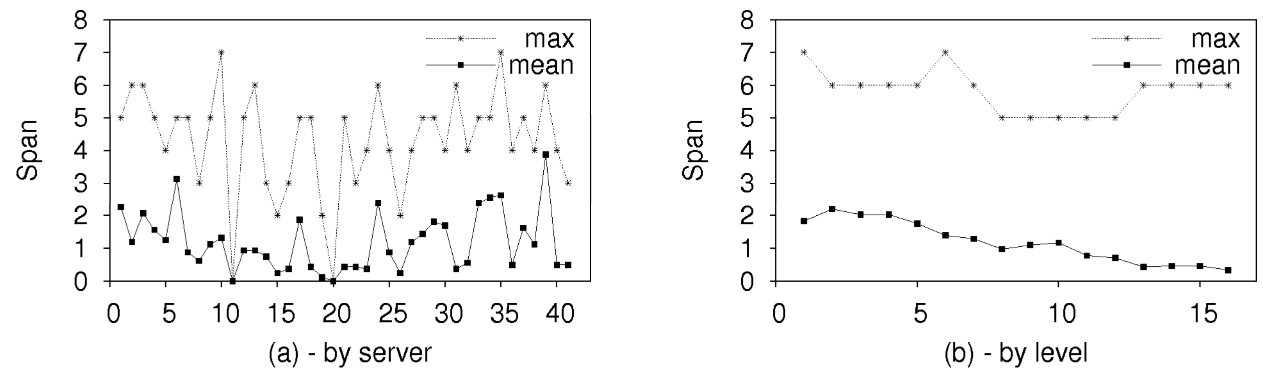


Figure 5: Span of periodicities in data set B ((a) - by server, (b) - by level)

Furthermore, measuring demand of individual applications would require instrumentation of the operating system, something which is hardly possible in a production data center.

The time series is a trace of the average CPU utilization across all CPUs in the server for a five minute interval ($D = 300$). The servers exhibit different CPU demand characteristics, which has consequences on the predictability of the signal. Figure 4 plots the CPU utilization of two exemplary servers, each over a one week period. Figure 4(a) shows a server with clear periodic behavior. The utilization is similar for the other weeks of data.

While the demand of the server in Figure 4(a) is seemingly predictable, Figure 4(b) shows an almost random demand behavior. Another group of five servers showed a very different behavior in the last $1/4$th of the trace data than in the first part. We removed these servers from the prediction testing.

5.2 Periodicity Mining Results

For the data set A we conducted tests to examine change-adaptivity in our algorithm. The number of levels was set to $L = 8$ and h to 0.5. We used $\tau_1 = 4800$ and $w_1 = 0.2$ (for pulse) as periodicity and width, resp. for the first part (share h) of the signal, and $\tau_2 = 6600$ and $w_2 = 0.5$ as the respective values for the second part. For both the sin-class and the pulse-class signal we varied the half-life T_h, looking at the values of 16 days and 4 days. For $T_h = 4$ days, newer periodic behavior ($\tau = 6600$ and $m_w = 0.5$) is captured correctly and the hit probability p_h reaches high values, but this is not the case for $T_h = 16$ days. This confirms that a non-adaptive approach (i.e. $T_h = \infty$) would be not able to capture the newer periodic behavior in this data set (unless only the most recent part of the trace would be used for training). Quality results are still produced up to the noise level of $p_n = 0.1$ for the sin-class and up to 0.15 for the pulse-class. The detailed result values for this study are left out due to space limits.

We performed periodicity mining for the data set B using the parameters $T_h = 7$ days and 32 levels. Of the 41 traces 36 featured periodicity in at least one level. The total number of levels with periodicities was 356. The most common values for τ were 15 minutes and 20 minutes. Relatively common were 1 hour, 1 day, and there are two occurrences of a 1 week periodicity. Altogether, the detected periodicities range from 15 minutes to 1 week.

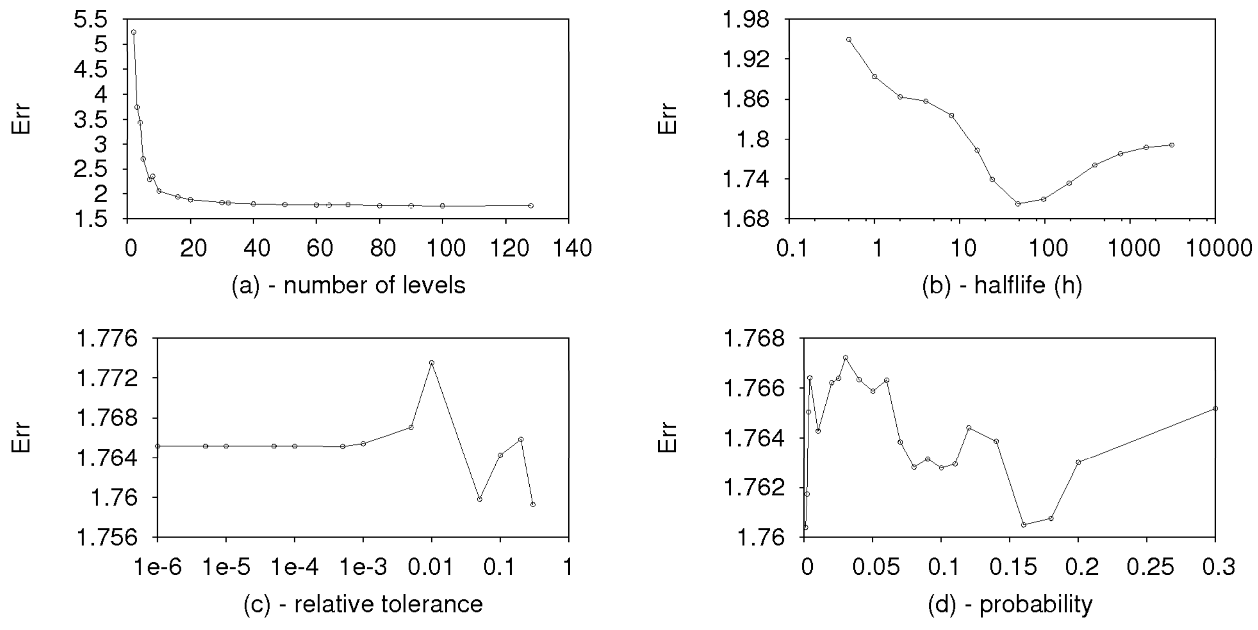


Figure 6: Influence of parameter values on the sum of the mean absolute errors over 31 servers of data set B ((a) - L, (b) - T_h, (c) - δ_r, (d) - p_{err})

In order to justify the change-adaptive approach we investigated how periodicities change within the data set B. We divided each trace into 8 sections of equal time lengths (using full trace, i.e. 100% of samples), and mined each section using $L = 16$ and $T_h = \infty$ (no change adaptivity). For each trace and each level we then counted the *section span of periodicities*: the number of consecutive sections having the same periodicity (in a particular level and trace). Intuitively, low span values indicate the need for change-adaptivity, since the periodicities do not last long. Figure 5(a) shows the means and maxima of spans over all 16 levels (by servers). For most servers, the section span average is below 2.0, or 1/4th of the trace duration, a value which affirms the positive impact of change-adaptivity for this data set. Figure 5(b) displays the means and maxima over servers (by levels). The span decreases with higher level index, which reflects the fact that the occurrences of the signal and corresponding periodic patterns are rare at higher values.

5.3 Prediction Testing

The test data cannot be directly compared with the predicted pmf's. We use two approaches to derive a value from a pmf: 1) we compute an expected value (*exp*), or 2) we take the median of a range belonging to a level with highest probability in the pmf (*topP*).

5.3.1 Parameter Sensitivity Analysis

We study the effects of essential parameters on the prediction accuracy using 31 servers from the data set B (servers having any periodicities and not being among the excluded five). The

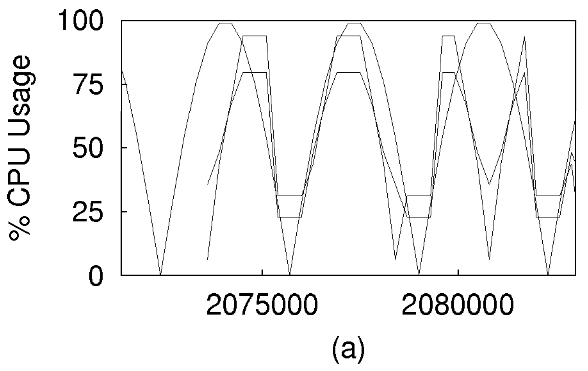
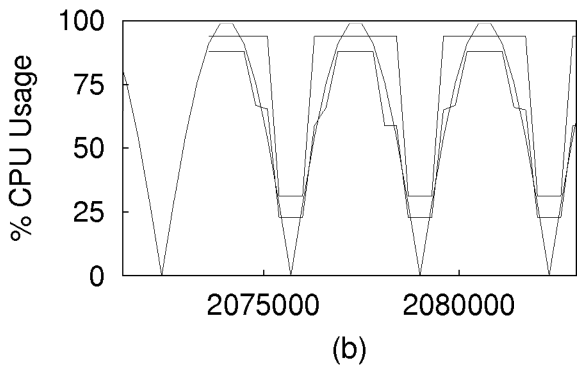


Figure 7: Prediction for synthetic data with $T_h = 16d$ (a) and with $T_h = 4d$ (b) (signal: solid; predicted pmf: the level with highest probability is dashed, the expectation is dotted; x-axis is time)



variable parameters are: the number of levels L, the half-life T_h, the relative tolerance δ_r, and the probability p_{err} that an accepted periodicity is due to a random phenomena only. We change one parameter at a time, while keeping the other three at default values. These values are $L = 128$, half-life $T_h = 28$ days, $\delta_r = 0.1$, and $p_{err} = 0.01$. The metrics for the prediction error Err are the sum of the mean absolute errors between the true signal values and the expectation of the predicted pmf (exp) over all 31 servers. Using maximum of the mean absolute error instead of the sum yields similar results.

The number of levels L has the largest influence on accuracy, as shown in Figure 6(a). Prediction accuracy increases with L, but saturates at values greater than 10. A well-suited value for data set B seems to be $L = 32$, which requires little memory for overhead data while offering reasonable prediction quality.

The influence of the half-life T_h on accuracy is worth noting, see Figure 6(b). Large values increase the error due to lack of adaptivity. However, too small a value for half-life impedes detection of longer-term periodicities, thus decreasing the accuracy. Finding a suitable value for T_h obviously depends on the "stability" of the periodic patterns and in most cases we need to take into account the idiosyncrasies of the application scenario. For data set B, $T_h = 2$ days produces the fewest errors.

As depicted by Figure 6(c) and (d), the relative tolerance δ_r and the probability p_{err} have marginal influence on the prediction accuracy and do not show any conclusive behaviour.

5.3.2 Prediction Results

Data Set A. Figure 7 illustrates the effect of change-adaptivity for the synthetic trace of type sin mined with half-life $T_h = 16$ days and $T_h = 4$ days (same scenario as in the periodicity mining experiment). Clearly a smaller half-life value (right figure) yields a more correct prediction due to adjustment to more recent periodicity, thus affirming our change-adaptivity approach.

Data Set B. Figure 8 shows prediction results for two typical servers. 8 levels are included on the left side, with 32 levels on the right. With an increasing number of levels the prediction

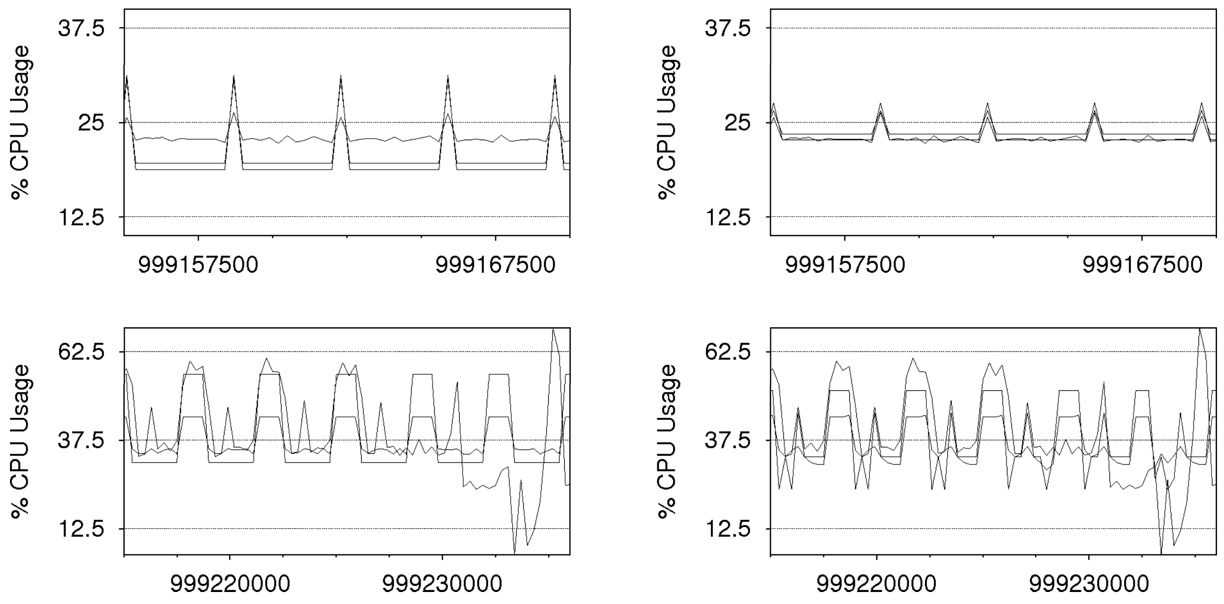


Figure 8: Prediction results for two server traces, with 8 levels (left) and 32 levels (right) each (signal: solid; predicted pmf: the level with highest probability is dashed, the expectation is dotted; x-axis is time)

follows the signal more closely, and more periodicities are discovered. As the expectation tends to average out the probabilities in a pmf, it takes less extreme values than the topP.

In Figure 9 we compare the prediction errors of our approach (a) with the same approach without change-adaptivity and (b) with the prediction based on the histogram-pmf explained below. We measure prediction errors in terms of the relative improvement of our method against the alternative. The relative improvement is computed according to the formula $100\,(err_{alt} - err_{period})/err_{period}$, where err_{alt} is the mean absolute error between the signal and the prediction based on the alternative method, and err_{period} is the mean absolute error between the signal and the periodicity-based prediction. Thus, positive values illustrate the advantage of our method.

Two variations were considered. The first variation measures errors only in cases when the signal is larger than the predicted value (*ut*). This corresponds to a situation in which the consumer gets less resources than needed, so that an under-provisioning occurs. This situations are more problematic since under-provisioning might lead to e.g. violation of Service Level Agreements or higher response times. The other variation measures the full error (*fd*), i.e. both over-provisioning and under-provisioning cases, and is more conservative.

The prediction approach based on histogram-pmf computes first the histogram-pmf of the signal (see Section 4.1). Then it derives the expected value (exp) and the level with highest probability (topP) from this pmf and uses them as the prediction result in all test cases, irrespective of the future time t of the prediction. This method represents a simple, yet change-adaptive prediction technique which we use as baseline for evaluating the prediction accuracy of our method.

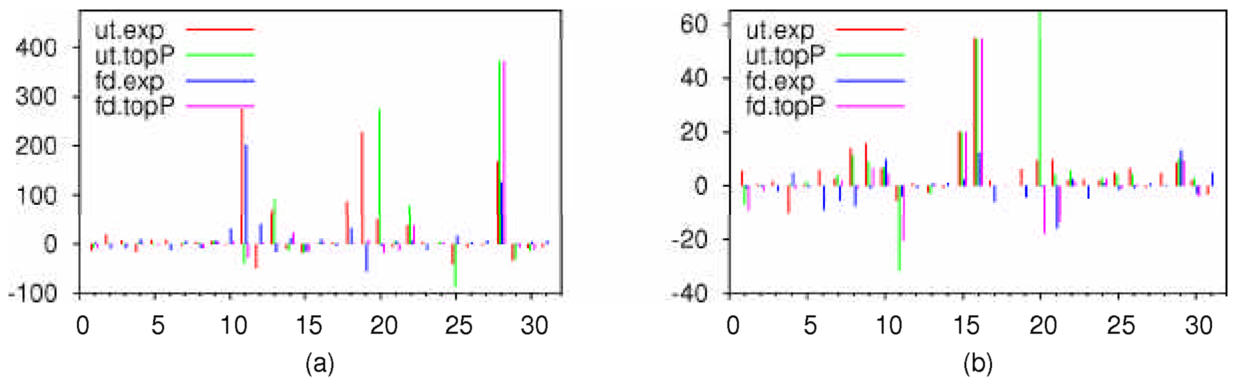


Figure 9: Relative improvement (in %) of the periodicity-based prediction compared to non-adaptive version (a) and the histogram-based prediction (b) for 31 server traces (x-axis is the server index)

We use $T_h = 2$ days and 32 levels as the parameter values. The merits of change-adaptivity of our method are shown in Figure 9(a). There is a very large accuracy improvement in several cases (up to 370%), however there are few instances with a small or middle accuracy degradation. The comparison against the (change-adaptive) histogram-pmf prediction in Figure 9(b) features a lot of cases with small improvement, but instances with large improvement are lacking (only in one case the improvement of ut.top exceeds 274% and is not shown in the diagram).

As for computational efficiency, the periodicity mining and prediction testing for all 31 servers (with approximately 2000 prediction tests in each case) required less than 130 seconds on a 2.5 GHz laptop with Java 1.5 (Sun JVM, Windows XP, Pentium 4).

6 Summary and Conclusions

We have described a change-adaptive method for characterizing the periodic behaviour in demand data, and derived a technique for long and short-term prediction of the demand. We placed special emphasis on low running time and compact output in the design of these algorithms, which are intended for use in tools for system self-management.

Our evaluation of the methods using both synthetic and data center traces clearly demonstrates the merits of the change-adaptive approach. Predictive accuracy on these traces is satisfactory, and the visual analysis shows that the predicted pmf's nicely capture trace characteristics. A more advanced test would utilize the predicted pmf as input to a scheduler such as that described in [11] and evaluate the obtained placements against ideal applications-to-servers mappings.

We intend to extend our work in several directions. We plan to combine our implementation with a GA-based scheduler to obtain an automated, proactive resource consolidation toolkit. We are also considering incorporating prediction techniques such as ARIMA to further improve the accuracy of our algorithm in short-term prediction. Further work can be done on defining non-equidistant levels. Implementation and comparison of alternative characterisation/prediction methods such as wavelet analysis is also planned. Finally, the characterization of periodic demand behaviour produced by our algorithms can serve as a basis for automated anomaly detection.

7 Acknowledgments

We would like to thank Minor Gordon and Ulf Hermann, ZIB Berlin, for help with editing this manuscript. Special thanks go to Jerry Rolia and Martin Arlitt from HP Labs Palo Alto for making the data set B available to us.

References

[1] G. Casella and R. L. Berger. Statistical Inference, 2nd edition, Brooks Cole, 2001.

[2] D. Dentscheva, W. Römisch. Optimal Power Generation under Uncertainty via Stochastic Programming. *Numerical Techniques and Engineering Applications (K. Marti, Ed.), Lecture Notes in Economics and Mathematical Systems.* Springer-Verlag, Berlin, 1997.

[3] P. A. Dinda. An Extensible Toolkit for Resource Prediction in Distributed Systems (RPS). http://www.cs.nwu.edu/~RPS/

[4] P. A. Dinda. Resource Signal Prediction and Its Application to Real-time Scheduling Advisors. *Ph.D. thesis*, CMU, 2000.

[5] W. Enders. Applied Econometric Time Series. 2nd Edition, Wiley Canada, 2003.

[6] J. Han, G. Dong, Y. Yin. Efficient Mining of Partial Periodic Patterns in Time Series Database. *Proc. 15th Int. Conf. on Data Engineering*, 1999.

[7] J. Hollingsworth, S. Maneewongvatana. Imprecise Calendars: An Approach to Scheduling Computational Grids. *Proc. Int. Conf. on Distributed Comp. Systems*, pages 352-359, 1999.

[8] HP Grid & Utility Computing. http://devresource.hp.com/drc/topics/utility_comp.jsp

[9] S. Ma, J. L. Hellerstein. Mining Partially Periodic Event Patterns With Unknown Periods, *ICDE 2001*, pages 205-214.

[10] Y. Qiao, J. Skicewicz, P. Dinda. Multiscale Predictability of Network Traffic. *Tech. Rep. NWU-CS-02-13.* Department of Computer Science, Northwestern University, 2002.

[11] J. Rolia, A. Andrzejak, M. Arlitt. Automating Enterprise Application Placement in Resource Utilities. *Proc. 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSOM 2003).* Heidelberg, Germany, October 2003.

[12] J. Rolia, X. Zhu, M. Arlitt, A. Andrzejak. Statistical Service Assurances for Applications in Utility Grid Environments. *Performance Evaluation Journal,* Vol. 58, No. 2-3, pages 319-339, Elsevier, Holland, 2004.

[13] L. W. Russell, S. P. Morgan, E.G. Chron. Clockwork: A New Movement in Autonomic Systems. *IBM Systems Journal - Vol. 42, No. 1*, 2003.

[14] J. A. Skicewicz, P. A. Dinda. Tsunami: A Wavelet Toolkit for Distributed Systems. *Tech. Rep. NWU-CS-03-16.* Department of Computer Science, Northwestern University, 2003.

[15] J. Yang, W. Wang, P. S. Yu. Discovering High Order Periodic Patterns. *Knowledge and Information Systems Journal (KAIS)*, 2004.

8 Biographies

Artur Andrzejak received the PhD degree in computer science from the Swiss Federal Institute of Technology (ETH Zurich) in 2000. He is currently a researcher at Zuse-Institute Berlin, Germany. He was a postdoctoral researcher at the Hewlett-Packard Labs in Palo Alto, California, from 2001 to 2002. His research interests include systems management and modeling, and Grids.

Mehmet Ceyran is working towards his Masters Degree in Computer Science at the Freie Universität Berlin, Germany. He is employed as a student programmer at Zuse-Institute Berlin since 2003. His research interests include software engineering, systems management and artificial intelligence.