Workload Analysis and Demand Prediction of Enterprise Data Center Applications

Daniel Gmach
Technische Universität München
85748 Garching, Germany
daniel.gmach@in.tum.de

Jerry Rolia and Ludmila Cherkasova
Hewlett-Packard Laboratories
Palo Alto, CA, USA
{jerry.rolia,lucy.cherkasova}@hp.com

Alfons Kemper
Technische Universität München
85748 Garching, Germany
alfons.kemper@in.tum.de

Abstract— **Advances in virtualization technology are enabling the creation of resource pools of servers that permit multiple application workloads to share each server in the pool. Understanding the nature of enterprise workloads is crucial to properly designing and provisioning current and future services in such pools. This paper considers issues of workload analysis, performance modeling, and capacity planning. Our goal is to automate the efficient use of resource pools when hosting large numbers of enterprise services. We use a trace based approach for capacity management that relies on *i)* the characterization of workload demand patterns, *ii)* the generation of synthetic workloads that predict future demands based on the patterns, and *iii)* a workload placement recommendation service. The accuracy of capacity planning predictions depends on our ability to characterize workload demand patterns, to recognize trends for expected changes in future demands, and to reflect business forecasts for otherwise unexpected changes in future demands. A workload analysis demonstrates the burstiness and repetitive nature of enterprise workloads. Workloads are automatically classified according to their periodic behavior. The similarity among repeated occurrences of patterns is evaluated. Synthetic workloads are generated from the patterns in a manner that maintains the periodic nature, burstiness, and trending behavior of the workloads. A case study involving six months of data for 139 enterprise applications is used to apply and evaluate the enterprise workload analysis and related capacity planning methods. The results show that when consolidating to 8 processor systems, we predicted future per-server required capacity to within one processor 95% of the time. The accuracy of predictions for required capacity suggests that such resource savings can be achieved with little risk.**

I. Introduction

In the distant past data centers were made up of small numbers of large mainframe computers that each hosted several application workloads with many users. Capacity planning experts helped to ensure that sufficient aggregate capacity was available just in time, as it was needed. With the advent of distributed computing new application workloads were typically assigned to their own smaller servers. The incremental cost of capacity from smaller servers was much less expensive than the incremental cost of capacity on mainframes. Capacity planners would often anticipate an application's workload demands two years in advance and pre-provision a new server with sufficient capacity so that the workload could grow into it. However, the explosive growth in both enterprise computing and Internet computing has led to server sprawl in data centers. Enterprise data centers are typically full of large numbers of lightly utilized servers that incur high cost of ownership including facilities cost, such as rent and power for computing and cooling, high software licensing cost, and high cost for human management activities.

Many enterprises are now beginning to exploit resource pools of servers supported by virtualization mechanisms that enable multiple application workloads to be hosted on each server. The primary motivation for enterprises to adopt such technologies is increased flexibility, the ability to quickly repurpose server capacity to better meet the needs of application workload owners, and to reduce overall costs of ownership. Unfortunately, the complexity of these environments presents additional management challenges. There are many workloads, a finite number can be hosted by each server, and each workload has capacity requirements that may frequently change based on business needs. Capacity management methods are not yet available to manage such pools in a cost effective manner.

The goal of our work is to provide a capacity management process for resource pools that lets capacity planners match supply and demand for resource capacity in a just in time manner. In this paper we characterize the workloads of enterprise applications to gain insights into their behavior. The insights support the development of capacity management services for the process.

We use a trace based approach for the capacity management services. The services implement *i)* the characterization of workload demand patterns, *ii)* the generation of synthetic workloads that predict future demands based on the patterns, and *iii)* a workload placement recommendation service. Our process automates data gathering and analysis steps that address these questions. As a result it enables human operators to handle the questions more quickly and accurately with lower labor costs.

To demonstrate the effectiveness of our proposed capacity management approach, we obtained six months of data from an enterprise data center. The data describes the time varying demands of 139 enterprise applications. We use the data to demonstrate the effectiveness of our approach. The results show that when consolidating to 8 processor systems, we predicted per-server required capacity to within one processor 95% of the time while enabling a 35% reduction in processor usage as compared to today's current best practice for workload placement. The remainder of the paper presents our results in more detail.

II. Capacity Management Process

This section describes the capacity management process we envision and its corresponding services. The process relies on a combination of sub-processes that implement various use cases for resource pool operators. Examples of use cases include:

- determine resource pool capacity needed to support a number of workloads;
- add/remove a workload to a resource pool;
- add/remove capacity to a resource pool;
- rebalance workloads across resources in a pool;
- reduce load on a server resource in a pool by recommending new workload placements for some of its workloads;
- report significant changes in workload demand behaviors; and,
- adjust per-workload forecasts, trends or quality of service requirements.

To support such use cases we must start with a definition of required capacity. *Required capacity* is the minimum amount of capacity needed to satisfy resource demands for workloads on a server resource [11]. Given a definition for required capacity, we implement:

- an admission control service;
- a workload placement service; and,
- a workload demand prediction service.

The *admission control service* decides whether a resource pool has sufficient resources to host a new workload. If so it reports which server the workload should be assigned to. We consider workloads that exploit multiple resources as a collection of individual workloads possibly having workload placement constraints that must be addressed by the workload placement service.

The *workload placement service* we employ recommends where to place application workloads among servers in the pool to reduce the number of servers used or to balance workloads across the servers. The service implements a trace based approach for characterizing resource demands and for recommending solutions. Basically, each workload is characterized using a time varying trace of demands for its key capacity attributes such as processor usage and memory usage. The workload placement service includes greedy algorithms for consolidating workloads onto a small set of servers and for balancing the workloads across some fixed number of servers. It also includes a genetic algorithm based optimizing search that aims to improve upon the greedy solutions. In each case the algorithms simulate multiple assignment scenarios. Each scenario considers the placement of zero or more workloads on each server. The aggregate demand of the workloads assigned to a server is characterized using a trace that is the sum of its per-workload time varying demands. The service recommends the best workload placement it can find over all servers, either for consolidation or for load leveling. Finally, *the service accepts additional constraints* on workload placements that include affinity between workloads, e. g., workloads must or must not be placed on the same physical server, and affinity between workloads and a list of one or more specific servers.

The *workload demand prediction service* has several purposes:

- it implements pattern discovery techniques;
- it helps to recognize whether a workload's demands change significantly over time;
- it supports the generation of synthetic traces that represent future demands for each workload to support capacity planning exercises; and,
- it provides a convenient model that can be used to support forecasting exercises.

The service is described in Section III.

The capacity management process further relies on the key concept of a *capacity management plan*. A capacity management plan is a calendar based data store that keeps track of: workload identities, forecasts, and resource access quality of service requirements; resources that are associated with a pool; and assignments of workloads to resources. As a calendar based data store, the plan keeps track of such information as a function of date and time. The information is used to support capacity planning.

III. Workload Demand Prediction

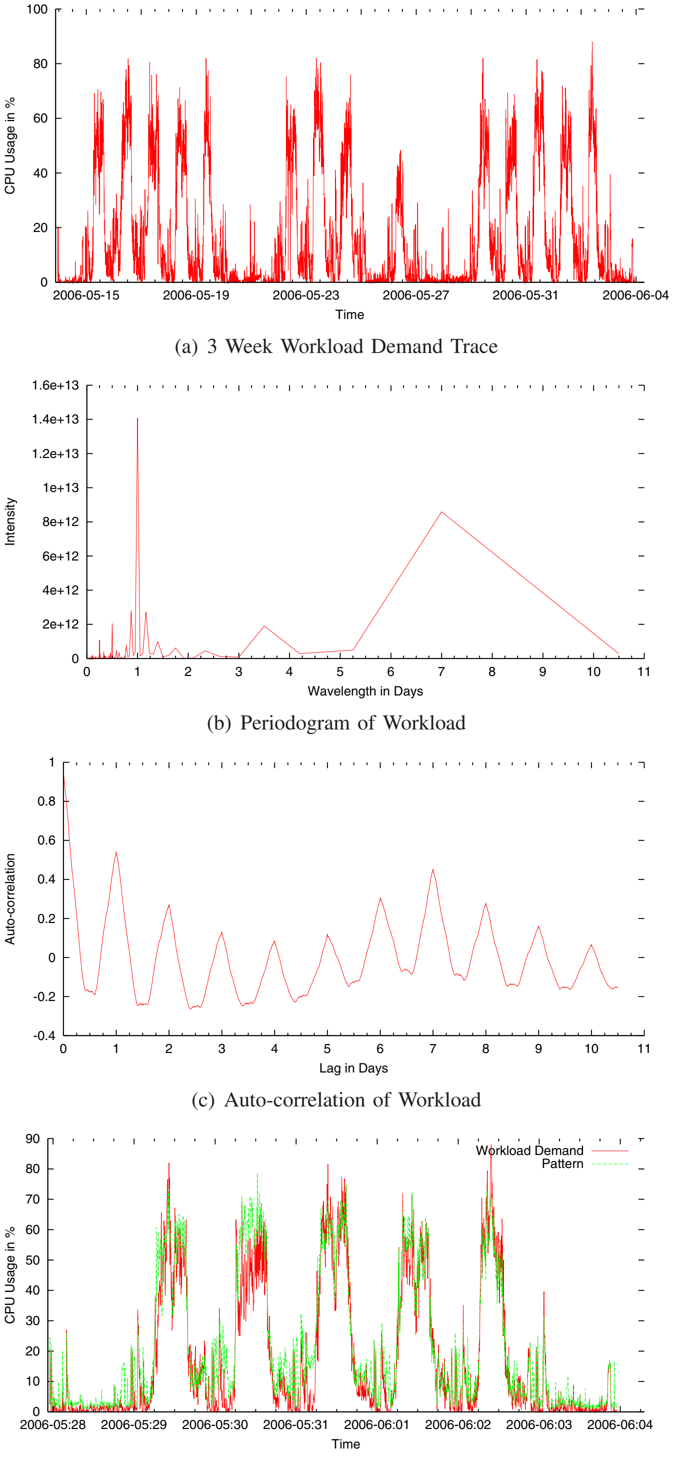
As stated in Section II, the workload demand prediction service has several purposes: *i)* to decide on a workload's demand pattern; *ii)* to recognize whether a workload's demands change significantly over time; *iii)* to support the generation of synthetic demand traces that represent future demands for each workload, e. g., demands for several weeks or months into the future, to support capacity planning exercises; and, *iv)* to provide a convenient model that can be used to support forecasting exercises. This section describes the techniques we used to implement this service.

A. Extracting Workload Patterns

We now present methods for deducing patterns, assessing their quality, and classifying them with regard to quality. A new approach is presented that assesses the similarity among occurrences of a pattern.

1) Pattern Analysis: Given a historic workload trace $L = (l(t_n))_{1 \leq n \leq N}$ which is represented by N contiguous demand values $l(t_n)$ we extract a demand pattern $P = (p(t_m))_{1 \leq m \leq M, M \leq N/2}$ with M contiguous demand values $p(t_m)$ with the assumption that the workload has a cyclic behavior. This assumption is evaluated later in the classification phase. According to a classical additive component model, a time series consists of a trend component, a cyclical component, and a remainder, e. g., characterizing the influence of noise. The trend is a monotonic function, modeling an overall upward or downward change in demand.

We illustrate our process for extracting a representative demand pattern from a workload using Figure 1. Figure 1(a), illustrates a three week workload demand trace with a public holiday during the second week.



(a) 3 Week Workload Demand Trace

(b) Periodogram of Workload

(c) Auto-correlation of Workload

(d) Extracted Pattern from Workload

Fig. 1. Extracted Pattern for Workload

To start the analysis, we identify the cyclical component that describes the periodic characteristics of the workload. To determine the yet unknown duration M of the pattern, we begin with an evaluation of the workload's *periodogram function* as shown in Figure 1(b). A Fourier transformation [7] gives an overlay of harmonics for the time-varying magnitude of demand. The periodogram shows the intensity I, with which a harmonic of a wavelength λ is present in the workload. The most dominant frequencies provide information about the duration of a potential pattern. Intuitively, if the periodogram function has a local maximum at $\lambda > 0$, then it is likely that there exists a representative pattern of length λ. In general, it is not the case that the wavelength with the global maximum, named $\max_I$, is most representative. Thus, we determine a set $\Lambda = \{\lambda_1, \ldots, \lambda_k\}$ of local maxima positions, with $I(\lambda_i) > \frac{\max_I}{2}$ for every $1 \leq i \leq k$. For instance, in the periodogram in Figure 1(b), we detect two local maxima. The first maximum proposes a wavelength of 1 day and the second maximum proposes one at 7 days.

In addition to the periodogram, we calculate the *auto-correlation* function for the workload demand trace. For a formal definition and further details on auto-correlation function see reference [3]. Figure 1(c) shows the auto-correlation function for the workload. It describes dependencies within the workload curve, i. e., the similarity between the workload and the workload shifted by a lag g. A high value ρ for the auto-correlation at lag g denotes that the workload curve shifted by g looks similar to the original one. Thus, if the auto-correlation shows local extrema at multiples of a lag g, it is a strong indicator that there exists a temporal dependency of length g. In the same way as with the periodogram, we determine a set of hypotheses $\{\lambda_{k+1}, \ldots, \lambda_{k+h}\}$ of significant local extreme positions and add them to the set Λ.

Workloads from enterprise data centers typically show a periodicity which is a multiple of hours, days, weeks, and so forth. Due to unavoidable computational inaccuracies and influences of irregular events and noise, the wavelengths in Λ can diverge slightly from these typical periods. We perform a comparison to calendar specific periods and determine for every wavelength candidate λ_i the best matching multiple of hours, days, and weeks and augment Λ with corrected wavelengths so that they are also considered.

In the second step, we select the best candidate wavelength λ' from the $\lambda_i \in \Lambda$. For each λ_i, we compute the average magnitude for ρ at multiples of λ_i. For example, if $\lambda_i = 1$ day, then we take the average of ρ_i from observations at lags of one day. If $\lambda_i = 7$ days, then we take the average of ρ_i from observations at lags of seven days. If the workload exhibits a pattern with length λ_i then the workload after shifting it by multiples of λ_i is similar to itself and thus the auto-correlation function exhibits high values at the lags $\{v \cdot \lambda_i \mid v \in N^+\}$. The average magnitude $\bar{\rho}_i$ is a measure of similarity among cyclic repetitions in demand for λ_i. For our example in Figure 1(c), $\lambda' = 7$ days has the highest average magnitude $\bar{\rho}'$ as compared to other values for λ_i and is recognized as the best pattern length. This implies that the pattern length is $M = 2016$ intervals[1] of duration $d = 5$ minutes. We note that the figure does not illustrate lags beyond 11 days even though they are included in the computation.

[1] There are 288 5-minutes intervals per day.

Total Minutes	**Week 1**	**Week 2**	**Week 3**
Week 1	-	181	69
Week 2	181	-	171
Week 3	69	171	-

TABLE I

Minutes per Day of Extreme Differences in Load Behavior

The chosen value for the pattern length of M intervals is used to calculate the pattern $P = (p(t_m))_{1 \leq m \leq M}$ for the workload. First we define occurrences for the pattern and then define the pattern's demand values $p(t_m)$. Given M, we divide the workload L into N/M complete occurrences and possibly one partial occurrence. Let O be the occurrences of the pattern for $o \leq N/M+1$. Thus, occurrence o is a subtrace of the trace L with values $l^o(t_m) = l(t_{m+o \cdot M})$ for each $1 \leq m \leq M$. For every interval t_m in the pattern we calculate a weighted average $p(t_m)$ for the interval. The weighted average is computed using intervals t_m from the occurrences O of the pattern. We define a weight for each occurrence o and interval m as:

$$w_{o,m} = \frac{l^o(t_m)}{\sum_o l^o(t_m)}$$

With these weights we compute the weighted average demand for each interval t_m as $p(t_m) = \sum_o w_{o,m} \cdot l^o(t_m)$. We use the weighted average to emphasize the importance of larger values over smaller values for capacity management.

Figure 1(d) shows the pattern and an occurrence of the pattern together in one diagram. The curves closely resemble one another.

2) Quality and Classification: The classification phase decides which workloads have periodic behavior. The classification is based on two measures for the quality of the pattern. The first measure is $\bar{\rho}'$ from Section III-A.1. Larger values for $\bar{\rho}'$ imply a better quality of fit. The second measure characterizes the difference between occurrences O and the pattern. The difference is computed as the average absolute error $\zeta = \frac{\sum_{1 \leq m \leq M, o} |p(t_m) - l^o(t_m)|}{N}$ between the original workload and the pattern P. Smaller differences suggest a better quality of pattern.

To classify the quality of patterns for a large number of workloads, we employ a *k means cluster algorithm* [10] with clustering attributes ζ and $\bar{\rho}'$. The algorithm partitions the patterns into three groups that we interpret as having strong, medium, or weak patterns. Weak patterns are not regarded as having a periodic pattern because no clear cycle could be deduced for the trace. This may be due to changes in workload behavior during the analyis period or because the pattern has a duration greater than half the analysis period.

3) Similarity of Behavior for Pattern Occurrences: We expect a certain amount of variation in demands among occurrences of a pattern. These may be due to random user behavior, holidays, etc.. However, larger variations may reflect a repurposing of a server or a change in business conditions that affect capacity management. We may choose to ignore atypical occurrences when estimating trends for demand or only use the most recent occurrences when estimating future workloads if demands have clearly changed. We now present an automated test to recognize whether there are significant differences between occurrences of a pattern.

The test is motivated by the Chi-square test [4]. It is designed to highlight extreme differences in load behavior. The test compares two occurrences at a time. For an occurrence o, we define a difference for time interval t_m as $p(t_m) - l^o(t_m)$. The differences for $1 \leq m \leq M$ express the variation of the occurrence o with respect to the pattern. We partition the difference values into three buckets. The three buckets have ranges $[-100, -10], (-10, 10], (10, 100]$, respectively. The differences in the range $(-10, 10]$ are deemed to be inconsequential from a resource pool capacity management perspective. The right and left buckets define the extreme differences from the pattern.

A Chi-square test can be used to determine whether a pair of occurrences, o and o', have statistically similar numbers of observations per bucket. However, we have found that interpreting the computed Chi-square statistic is problematic. The value of the statistic is sensitive to the number of observations in the right and left buckets and the interpretation of the value depends on pattern lengths. Instead, we choose to consider the sum of the absolute differences in counts for the left and right buckets. This sum tells us whether the occurrences differ from the pattern in a similar way. The sum is a count of intervals and can be expressed in terms of the number of minutes per day that the occurrences have differences in extreme behavior.

Table I gives the resulting minutes per day differences in extreme load behavior as computed for the workload in Figure 1(a). Weeks 1 and 3 have differences in extreme behavior of approximately 69 minutes per day. Week 2 differs from the other weeks. It has differences in extreme behavior of 181 and 171 minutes per day as compared with week 1 and week 3, respectively. This is likely due to the holiday that occurred in week 2. In the case study we consider the impact of alternative values for a threshold that decides whether a pair of occurrences differs significantly in behavior.

B. Analyzing the Trend

To characterize a trend of the workload we calculate the aggregate demand difference of each occurrence of the pattern from the original workload L. Let c^o_m be the difference between the $p(t_m)$ and the demand value for interval t_m in the occurrence o. We define c^o as the aggregate demand difference of occurrence o with respect to the pattern P as: $c^o = \sum_{1 \leq m \leq M}(p(t_m) - l^o(t_m))$. Further, we define the trend τ as the gradient of the linear least squares fit [8] through the values c^o for the occurrences O as ordered by time. The trend τ estimates the change of demand over time with respect to the pattern.

C. Generating Synthetic Workload Traces and Forecasting

We now consider a process for generating a synthetic trace to represent a future workload demand trace L' for some time period in the future. Typically, we generate traces to represent demands for a time period that is several weeks or months into the future.

Our goal for a synthetic trace is to capture the *highs* and *lows* of demand and *contiguous sequences* of demand. These are critical for modeling a workload's ability to share resource capacity with other workloads and to model required capacity for the workload. Furthermore, our approach must be able to introduce an observed trend or forecast information.

To generate an occurrence o' for L' we rely on the historical pattern occurrences O. A value $l^{o'}(t_m)$ is chosen randomly from the corresponding t_m values from O. Given a sufficiently large number of future occurrences O', we will obtain the same time varying distribution of demands as in O. This gives us a pattern of demands that captures the lows and highs of demand in a representative way. Furthermore, we note that the occurrences may have a trend τ. For the sequence of historical pattern occurrences we normalize the demand values so that the trend is removed with respect to the last occurrence before constructing O'. This allows us to forecast demands for synthetic traces based on τ and time into the future.

Demands $l^{o'}(t_m)$ in the synthetic trace are augmented to reflect the trend τ. We assume an additive model. For each future occurrence o', we compute an absolute value based on τ that must be added to each demand in occurrence o'. The further o' is into the future the greater the change with respect to the historical data, assuming τ is not zero.

To better model burstiness in demand we must take into account sequences of contiguous demands in the trace L. We accomplish this by randomly selecting blocks of b intervals $t_m, t_{m+1}, \ldots, t_{m+b}$ at a time from the occurrences O. In this way, the synthetically generated traces have contiguous sequences of demand that are similar to the historical trace.

In our capacity management process, we repeat our analysis steps using multiple randomly generated instances of L' to better characterize the range of potential behavior for the overall system. Multiple instances better characterize interactions in demands among multiple workloads.

Finally, a workload pattern P provides a convenient way to express what-if-scenarios and business forecasts that are not observable to us from historic data. Suppose we have a pattern P with O occurrences and we require a change to the pattern. Then, we can express a change once with respect to P rather than once for each of the possibly many occurrences.

IV. Case Study

To evaluate the effectiveness of our methods and processes we obtained six months of workload trace data for 139 workloads from a data center. The data center specializes in hosting enterprise applications such as customer relationship management applications for small and medium sized businesses. Each workload was hosted on its own server so we use resource demand measurements for a server to characterize the workload's demand trace. The measurements were originally recorded using vmstat and sar. Each trace describes resource usage, e. g., processor and memory demands, as measured every 5 minutes starting January 1st, 2006. Our case study considers:

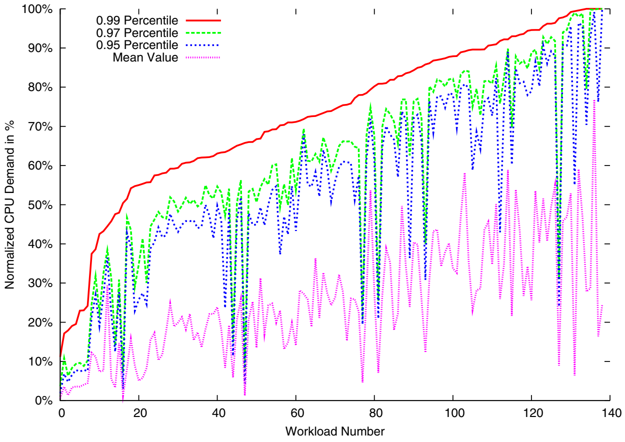


Fig. 2. Top Percentile of CPU Demand for Applications under Study.

- a characterization of the data center's workloads;
- results from workload demand pattern analysis;
- an analysis of similarity among occurrences of patterns;
- a validation of the trending and synthetic workload generation techniques; and
- a walk-forward test that employs the pattern matching, trending, and synthetic workload generation methods.

A. Workload Characterization

This section illustrates the nature of the enterprise application workloads under study. We show percentiles of demands and durations for bursts of demands. Figure 2 gives the percentiles of CPU demand for the 139 applications over a period of 5 weeks. We chose to limit the duration to 5 weeks so that we didn't exaggerate the peak demands beyond what we may use as part of the proposed capacity management process. The demands we illustrate are normalized as a percentage with respect to their peak values. Several curves are shown that illustrate the 99th, 97th, and 95th percentile of demand as well as the mean demand (the workloads are ordered by the 99th percentile for clarity). The figure shows that more than half of all studied workloads have a small percentage of points that are very large with respect to their remaining demands. The left-most 60 workloads have their top 3% of demand values between 10 and 2 times higher than the remaining demands in the trace. Furthermore, more than half of the workloads observe a mean demand less than 30% of the peak demand. These curves show the bursty nature of demands for most of the enterprise applications under study. Consolidating such bursty workloads onto a smaller number of more powerful servers is likely to reduce the capacity needed to support the workloads.

An additional and complementary property for a workload is the maximum duration of its contiguous application demands. While a system must be provisioned to handle sustained bursts of high demand, short bursts may not significantly affect the workload's users. For example, if a workload's contiguous

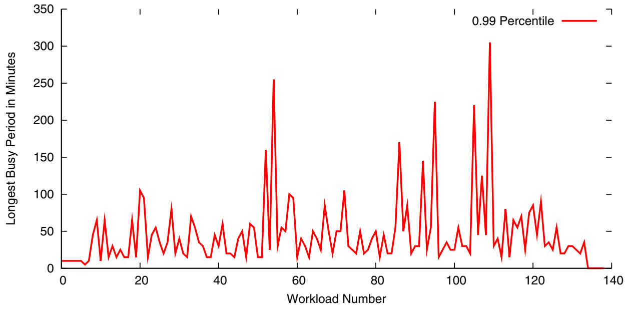


Fig. 3. Max Duration of the Bursts above 99-Percentile of Demand for Applications under Study.

demands above the 99th percentile of demand are never longer than 10 minutes then it may be economical to support the workload's 99th percentile of demand and allow the remaining bursts to be served with degraded performance [6].

Figure 3 presents the maximum duration of the contiguous demands above 99th percentile of the workload demand. The figure shows that for 50% of the workloads the periods of a high load are very time-limited:

- 23.7% of the workloads have a longest busy period less than 15 minutes;
- 34.5% of the workloads have a longest busy period less than 20 minutes;
- 53.3% of the workloads have a longest busy period less than 30 minutes.

Therefore, for a significant portion of the enterprise applications under study, allowing a time-limited degraded application performance (e. g., up to 30 min.) is likely to offer significant savings in the amount of capacity that must be provisioned.

B. Workload Pattern Analysis

This section presents general results for the workload pattern analysis. The results we present consider workload demand traces from April, 1st 2006 to July, 8th 2006. To begin we offer a general overview of the workloads. Figure 4 gives a summary of the pattern lengths for the 139 workloads. The pattern analysis extracts patterns with lengths between three hours and seven weeks:

- 68% of the workloads exhibit a weekly behavior, and
- 17% of the workloads exhibit a daily behavior.

We note that not all of the pattern lengths are directly related to a multiple of days, for example one workload exhibits a strong cyclical behavior with a period of 10 days, 10 hours, and 45 minutes. Thus having knowledge of the patterns can help to recognize when workloads with different pattern durations will have collisions for their larger demands.

Using the clustering algorithm, we classified the 139 patterns in the following way. There were

- 31 *strong* patterns. Most of the 31 strong patterns correspond to batch jobs that exhibit a very distinct cyclic behavior;
- 76 *medium* patterns. The medium patterns typically include interactive and/or mixed batch and interactive work;

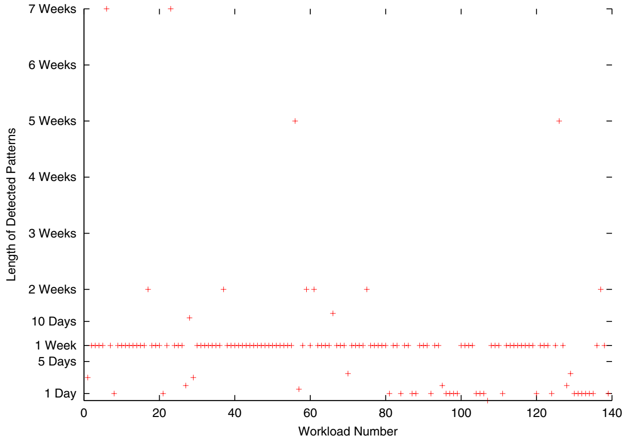


Fig. 4. Lengths of Workload Demand Patterns

- 32 *weak* patterns. The weak patterns include: *i)* workloads showing no cyclic behavior, e. g., constant or random demands, *ii)* workloads that have been interrupted several times, e. g., by intermediate peaks with 100% load each lasting a couple of days, or *iii)* workloads that changed completely during the duration of the workload trace, e. g., the workload in Figure 5.

These results suggest that pattern matching methods deduce reasonable patterns for 107 out of 139 cases.

C. Similarity of Behavior for Pattern Occurrences

As discussed in Section III-A.3, we need to understand when there are significant differences in a workload's pattern occurrences. Significant differences may cause a pattern to be classified as weak.

Figure 5 shows a 14 week workload demand trace for a workload classified as having a weak pattern. There is a clear discontinuity in behavior at week 10 and what appear to be three separate patterns.

The pattern chosen for this workload is influenced heavily by the first 8 weeks of the workload. Figure 6 shows a plus-minus CDF for variability of differences in demand with respect to the overall pattern for each of the 14 weeks. The figure shows that there are large differences in the tails of the differences in demand with respect to the pattern. Table II shows the range of minute per day differences in extreme behavior for the occurrences with respect to week 1. The table shows that weeks 1 through 8 have average differences of approximately an hour per day – except for week three which has a difference of 109 minutes per day, while the others have

	Week 2 – 8	Week 9	Week 10	Week 11 – 14
Week 1	36 – 66 (109 for week 3)	241	817	302 – 630

TABLE II
RANGE OF MINUTES PER DAY OF DIFFERENCES IN EXTREME LOAD BEHAVIOR

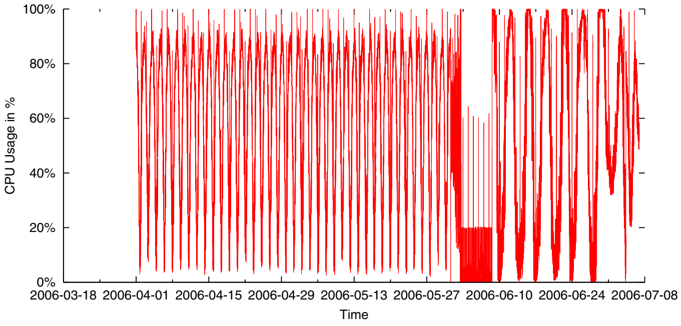


Fig. 5. Changes to the Workload's Demands

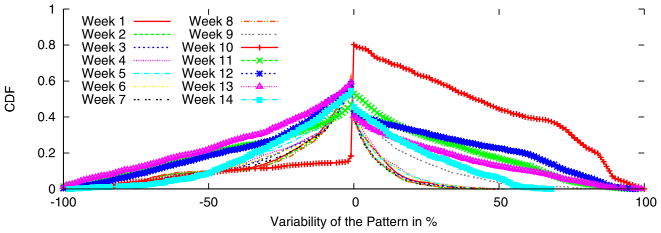


Fig. 6. Variability of Differences in Demand with Respect to Pattern

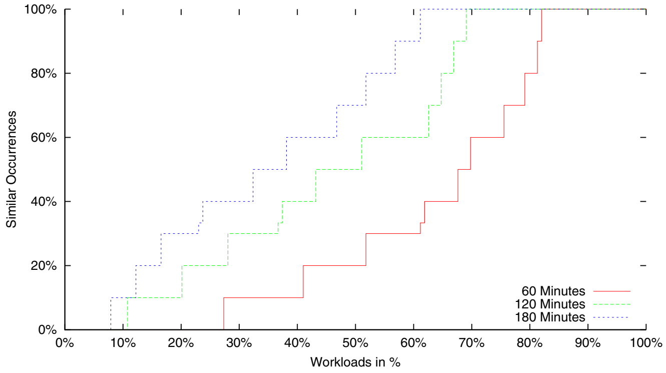


Fig. 7. CDF of Differences in Extreme Behavior in Minutes per Day

differences of 4 or more hours per day. Thus the approach we present is able to distinguish such changes in demands across occurrences and can provide insights into why some patterns are classified as weak.

Figure 7 considers all 139 workloads for a five week analysis period. Five weeks is a typical period for our use of these methods. It shows the percentage of workloads that have a fraction of occurrences pairs with differences in extreme behavior of less than 60, 120, and 180 minutes per day, respectively. We see that the 120 minute per day scenario has 31% of workloads where all occurrences are similar, and 20% of workloads where no more than 20% of occurrences are similar. This corresponds well to the breakdown of pattern quality we observed from the clustering algorithm of Section IV-B for these same 5 weeks. The clustering algorithm had 24% strong patterns and 19% weak patterns. We note that our new approach lets us classify the quality of a patttern on a per-workload basis, i.e., without the need for clustering. For the 60 minute threshold, only the top 30% of workloads have more than 50% of occurrence pairs being similar. As expected the 60 minute threshold is more strict causing more pairs of occurrences to be regarded as dissimilar. Likewise the 180 minute threshold is less restrictive. We choose to use the 120 minute threshold because it has a good correspondence with the clustering based classification system.

D. *Trending*

The approach to trending that we employ assumes an additive model. Historic data is used to estimate the expected change in demand from one occurrence to the next. This change is applied repeatedly when generating the synthetic traces for future occurrences. There are a few challenges that arise when applying this method. This section discusses two of them, and how we address them.

First, public holidays, runaway operating system processes, and failed operating system processes may each influence what is perceived as a trend. Long term trends are less affected by these events. Pattern similarity can warn of occurrences that deviate from the observed pattern, and such occurrences can be excluded from a trend.

Secondly, sufficient historical data is needed to predict a trend for a period of time into the future. For example, a minimum of two weeks of data are needed to predict a weekly trend. However, short term trends, e. g., on the order of days or weeks, may exist that are not representative of the longer term. For example, the last week of a month may always have greater demands than the first three weeks. Depending on where the historic data starts, trending methods may identify an increasing or decreasing short term trend. These trends exist but each has a particular time into the future for which it is relevant. Significant historical data is needed to capture trends that are on the timescale of quarter years. At these longer timescales applications demands may change, due to new application functionality or software releases, or business conditions may change thereby making such trends less useful. For long timescales business forecasts aim to capture such disruptions. They must be represented in the capacity plan.

With knowledge of the above limitations, we can still exploit trending for shorter timescales in the capacity management process. Figure 8 shows a workload along with a trend τ that we compute using three weeks of historical data. The figure shows a slowly decreasing trend of -2.3 units of demand per week that correctly anticipates decreasing demands one or two weeks into the future.

E. *Representativeness of a Synthetic Trace with Trending*

In this section, we illustrate the representativeness of a synthetic workload trace generated using our approach. We use three weeks of historic data from May 14 through June 4 to generate a synthetic trace for the next two weeks, using trending, and compare the characteristics with that of the actual workload data for the following two weeks, namely June 5 through June 18. Figure 8 shows the corresponding

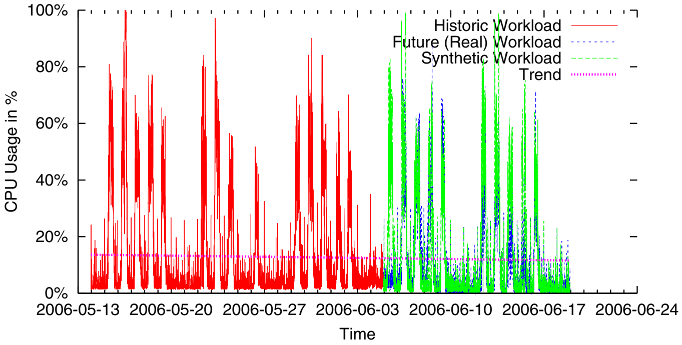


Fig. 8. Trending

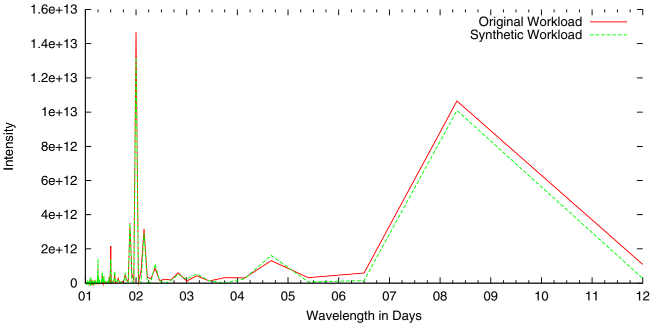


Fig. 9. Comparing Periodogram of Synthetic and Original Workload

historical, synthetic, and future workload demands. The trend is clearly useful at this timescale.

To further assess the representativeness of the synthetic trace as compared with the historic data, we see that Figures 9 and 10 show that the periodogram and auto-correlation functions for the two data sets are very similar. Finally, the required capacity values for the historic, synthetic, and actual future workload demands were 508, 455, and 461 units of demand, respectively. The corresponding contiguous bursts of demand that were beyond the 99-percentile were 160, 60, and 35 minutes, respectively. Thus the synthetic trace has both a similar pattern and required capacity as the actual demand trace it aimed to predict. In the next section, we further validate aspects of this workload demand prediction service.

F. Walk-Forward Test

In this section, we exploit the workload demand prediction service as part of the capacity management process. We conduct a walk-forward test over the six months of data to emulate how well our capacity management process would have served the data center for the six months.

- Starting with the first week, a window with w weeks of data is used to recommend a consolidated configuration C_1, i. e., each workload is assigned to a specific server, for the system. The configuration reports expected capacity values for each server in the configuration. Multiple synthetic traces, in our case thirty, are used to determine a range of estimates for required capacities for each server. The greatest observed required capacity for a server is chosen as the estimate for required capacity of the server.

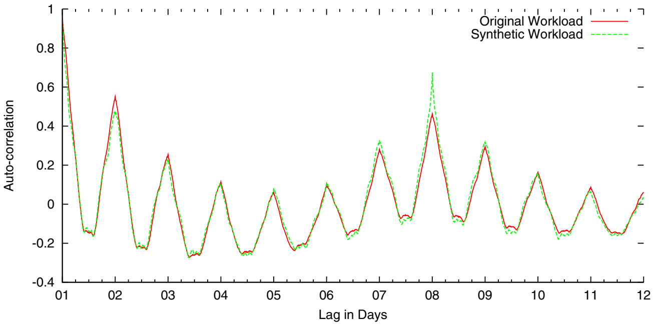


Fig. 10. Comparing Auto-correlation of Synthetic and Original Workload

- The next y weeks of data are then simulated with respect to C_1. This simulation gives the actual capacity for the next y weeks.
- The difference between a server's estimated and actual capacity gives the absolute error for the estimate of capacity. The negative errors reflect "under-estimated" capacity while the positive errors correspond to "over-estimated" capacity. We use a plus-minus CDF that reflects both types of errors for the walk-forward test.
- The steps in the walk-forward test are repeated iteratively with w weeks of data but now starting with weeks 2, 3, and so on.
- Let i be the step number in the walk-forward test. Step i computes a new configuration C_i and a new set of differences between estimated and actual required capacity values for each server.

We consider an ongoing process where the workloads are repeatedly consolidated onto a number of powerful servers over time. The servers have 8 processors. In general, the consolidation required 13 to 15 of these servers at a time. To evaluate the effectiveness of workload demand prediction methods we consider several different scenarios for generating synthetic workloads. The scenarios include:

a) use pattern analysis and trending;
b) use pattern analysis alone;
c) all workloads are associated with daily pattern; and,
d) all workloads are associated with a 30 hour pattern (specifically chosen to be incorrect).

For our study we use $w = 5$ weeks of historic input for the process and predict required capacity $y = 1$ week and $y = 5$ weeks into the future. Figures 11 and 12 show CDFs of errors in predictions for required capacity for the scenarios over the entire walk-forward test. A negative error suggests that a method estimates less capacity than is actually required for a server.

Figure 11 shows the results for the one week prediction. Scenarios *a)* and *b)* are pretty much indistinguishable. Trending avoided two large but similar negative errors. A fixed daily pattern without trending, scenario *c)*, caused several larger negative errors than *a)*, i. e., values less than -1 processor. The clearly incorrect 30 hour pattern from scenario *d)* caused severe errors.

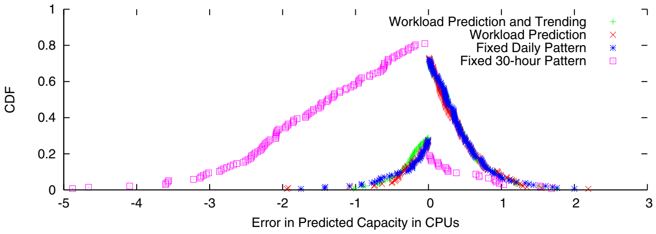


Fig. 11. Predicting Capacity for 1 Week

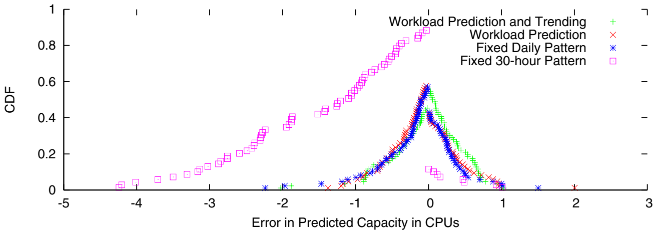


Fig. 12. Predicting Capacity for 5 Weeks

Figure 12 shows that the results for predicting required capacity 5 weeks into the future are very similar. The only difference is errors were a little lower for scenario *b)*, i. e., without trending, than *a)* with trending. This is reasonable. Our historic window of 5 weeks of data is not likely to be sufficient for predicting trends 5 weeks into the future for all workloads for all steps in the walk-forward test.

For both 1 week and 5 week predictions, Scenario *a)* estimates per-server required capacity to within one processor (out of eight processors) 95% of the time.

V. Related Work

Existing studies of internet and media workloads [1], [5] indicate that client demands are highly variable (“peak-to-mean” ratios may be an order of magnitude or more), and that it is not economical to overprovision the system using “peak” demands. However, we are not aware of similar studies for enterprise workloads. We present results that illustrate the peak-to-mean behavior for 139 enterprise application workloads. An understanding of burstiness for enterprise workloads can help to choosing the right trade off between the application quality of service and resource pool capacity requirements. The ability to plan and operate at the most cost effective capacity is a critical competitive advantage.

Historically, enterprise capacity management groups have relied upon curve fitting and queueing models to anticipate capacity requirements for shared resources such as mainframes or large servers. Curve fitting and business level demand forecasting methods are used to extrapolate measurements of application demands on each resource. Queueing models may be used to relate desired mean response times for model specific workload classes [14], [15], [16] (e. g., batch or interactive, payroll, accounts receivable) to target for maximum resource utilizations. Unfortunately, this approach is typically a people intensive and hence expensive process. Our approach accounts for the non-linear relationship between utilization and application responsiveness by considering the relationship between resource demand and the utilization of time-varying capacity as allocated by workload managers [6].

We believe that understanding workload patterns, trends, and forecasts, and using them for future demand prediction is critical for a capacity management process that aims to make efficient use of capacity for resource pools. The demand prediction we consider predicts demands days and weeks into the future. We distinguish the methods we employ from those that are typically used to predict demands several seconds or minutes into the future. Techniques for very short term predictions often use other approaches such as ARMA [3] or GARCH [9], [2] models. While these approaches may be appropriate for the very short term their predictions quickly converge to a mean value for the time scales of interest to us. [18] also describes methods for predicting workload demand patterns that exploit periodograms and auto-correlation. They are similar to the methods we propose, but do not consider trends, or synthetic workload generation as we developed in this paper.

Traces have been used to support what-if analysis that consider the assignment of workloads to consolidated servers. VMware Capacity Planner [17] and TeamQuest [13] offer products that employ trace-based methods to support consolidation exercises. AutoGlobe [12] proposes a self-organizing infrastructure where the available hardware is virtualized, pooled, and monitored. They introduce a fuzzy logic based controller to supervise all services running on the hardware platform. If the controller recognizes an exceptional situation it triggers actions to remedy the situation automatically. In addition to that, they introduce a static optimization module that uses historical demand information to compute workload placement recommendations. They calculate the recommendation using a greedy heuristic.

We believe the workload placement service we employ has advantages over other workload placement services described above. It supports both consolidation and load balancing services as needed in a comprehensive capacity management process and is supported by a genetic algorithm that tends to improve upon greedy workload placement solutions. Furthermore, the workload placement methods go further than the other methods by addressing per workload resource access quality of service specifications, classes of service, and placement constraints.

VI. Conclusions and Future Work

We describe a capacity management process for resource pools. The process relies on services that automate and simplify management for resource pool operators. In this paper we focused on a workload demand prediction technique. A case study exploited six months of data for 139 enterprise applications to evaluate the effectiveness of our methods. The automated methods predicted the capacity of servers hosting the workloads to within one processor out of eight 95% of the time while reducing aggregate processor requirements by 35% without significant risks. Such advance knowledge can help resource pool operators to decide whether to order additional capacity for their pools.

The workload demand prediction service relies on pattern and trend recognition methods. We characterized the quality of models for workload patterns and introduced a method to automatically recognize when occurrences of a pattern have large differences in behavior. The technique recognizes transitions in behavior that cause poor quality patterns. We introduced a method to generate synthetic traces to represent the future behavior of workloads, taking into account medium term trends, e. g., weekly. The synthetic trace had similar properties as the actual future behavior of the workload. Our case study results show that trend prediction can be helpful as long as we do not exaggerate how far into the future we expect trends to continue. We believe that workload demand prediction methods are advantageous for a capacity management process. They recognize odd patterns which may interact in the future, e. g., 3 day and 5 day patterns may interact every 15 days, and can help to report when a workload's demands appear to deviate from past behavior.

Our future work includes: developing on-line automated methods for monitoring and reporting unplanned changes to workload characteristics; better exploiting notions of confidence and risk regarding predictions for future capacity requirements; and better integrating business forecasts for change into our approach.

References

[1] M. Arlitt and C. Williamson. Web Server Workload Characterization: The Search for Invariants. In Proc. of the ACM SIGMETRICS '96 Conference, Philadelphia, PA , May 1996.

[2] T. Bollerslev. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 31:307–327, 1986.

[3] G. E. P. Box, G. Jenkins and G. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, Upper Saddle River, NJ, USA, third edition, 1994.

[4] I. Chakravarti, R. Laha, J. Roy.Handbook of Methods of Applied Statistics, vol. I, John Wiley and Sons, pp. 392-394, 1967.

[5] L. Cherkasova and M. Gupta. Characterizing Locality, Evolution, and Life Span of Accesses in Enterprise Media Server Workloads. In *Proceedings of NOSSDAV*, May 2002.

[6] L. Cherkasova and J. Rolia: R-Opus: A Composite Framework for Application Performability and QoS in Shared Resource Pools. Proc. of the Intl. Conference on Dependable Systems and Networks (DSN'2006), Philadelphia, PA, USA, June 2006.

[7] J. W. Cooley and J. W. Tukey. An Algorithm for the Machine Computation of Complex Fourier Series. In *Mathematics of Computation*, vol. 19, pages 297–301, 1965.

[8] N. R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley & Sons, New York, NY, USA, third edition, 1998.

[9] R. F. Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50(4):987–1008, 1982.

[10] J .A. Hartigan and M. A. Wong. A K-Means Clustering Algorithm. In *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[11] J. Rolia, L. Cherkasova, M. Arlitt and A. Andrzejak. A Capacity Management Service for Resource Pools. Proc. of the 5th Intl. Workshop on Software and Performance (WOSP'2005), Spain, 2005.

[12] S. Seltzsam, D. Gmach, S. Krompass and A. Kemper. AutoGlobe: An Automatic Administration Concept for Service-Oriented Database Applications. Proc. of the 22nd Intl. Conference on Data Engineering (ICDE'2006), Industrial Track, Atlanta, GA, 2006.

[13] TeamQuest. `http://www.teamQuest.com`

[14] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal. Dynamic Provisioning of Multi-tier Internet Applications. Proc. of the 2nd IEEE Intl. Conference on Autonomic Computing (ICAC-05), Seattle, 2005.

[15] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. An Analytical Model for Multi-tier Internet Services and its Applications. Proc. of the ACM SIGMETRICS'2005, Banff, Canada, June 2005.

[16] D. Villela, P. Pradhan, D. Rubenstein. Provisioning Servers in the Application Tier for E-Commerce Systems. Proc. of IWQoS'04, Montreal, Canada, 2004.

[17] VMware Capacity Planner. `http://www.vmware.com`

[18] M. Wimmer, V. Nicolescu, D. Gmach, M. Mohr, A. Kemper and H. Krcmar. Evaluation of Adaptive Computing Concepts for Classical ERP Systems and Enterprise Services. *Proc. of the Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services (CEC'06 and EEE'06)*, San Francisco, CA, 2006.