Towards Workload-Aware Self-Management: Predicting Significant Workload Shifts

Marc Holze #1, Ali Haschimi #2, Norbert Ritter #3

#Department of Informatics, University of Hamburg
Vogt-Kölln-Str. 30, Hamburg, Germany
[1]holze@informatik.uni-hamburg.de
[2]2haschem@informatik.uni-hamburg.de
[3]ritter@informatik.uni-hamburg.de

***Abstract*— The workloads of enterprise DBS often show periodic patterns, e.g. because there are mainly OLTP transactions during day-time and analysis operations at night. However, current DBS self-management functions do not consider these periodic patterns in their analysis. Instead, they either adapt the DBS configuration to an overall "average" workload, or they reactively try to adapt the DBS configuration after every periodic change as if the workload had never been observed before.**

In this paper we propose a periodicity detection approach, which allows the prediction of workload changes for DBS self-management functions. For this purpose, we first describe how recurring DBS workloads, i.e. workloads that are similar to workloads that have been observed in the past, can be identified. We then propose two different approaches for detecting periodic patterns in the history of recurring DBS workloads. Finally we show how this knowledge on periodic patterns can be used to predict workload changes, and how it can be adapted to changes in the periodic patterns over time.

I. Introduction

The workload of a real-world DBS is typically not stable over time, but changes as new applications are installed or new users appear, for example. Whenever there is such an evolution in the workload, the configuration of the DBS must be adapted accordingly. Formerly, all configuration changes had to be identified by a database administrator (DBA). In order to reduce the maintenance costs of DBS, the DBMS vendors have recently equipped their products with a rich set of self-management functions.

In addition to irregular, permanent changes to the workload, DBS often also face periodic changes. For example, the workload in a DBS used for data warehousing significantly differs between day-time, when complex queries are processed, and night-time, when bulk-updates are executed. In this case, the DBS performance would significantly benefit from two distinct configurations: indexes, large sort areas, and a high optimization level at day-time, and no indexes, utility function heap space and a low number of parallel users at night-time. Of course, the periodic patterns may also be more complex. For example, there may be OLTP workloads in the morning and afternoon hours, whereas reporting queries are executed at noon and in the evening, and batch updates at night.

However, these periodic workload changes are currently not considered by existing self-management functions. Off-line tools like index advisors [1],[2] or configuration advisors [3] do not consider changes in the workload at all, but the DBA has to manually re-execute them when he suspects a benefit. In contrast, on-line self-management functions automatically adapt the DBS to usage changes. However, many of these functions (e.g. automated memory managers [4], [5]) learn a model of the observed system behaviour. As this model is valid for a specific workload only, it will produce incorrect results when the workload changes. Hence, several hours may be required to learn a new model and to adapt the DBS to it. By then, the workload may already have changed back to the old pattern again. Although the adaptation to workload changes in other on-line self-management functions may be faster (e.g. index selection [6]), they do not consider workload periodicity explicitly. Hence, they may for example trigger the costly creation of a new index, although the current workload will most probably be replaced soon.

In this paper we present an approach for the identification of periodic DBS workloads. It is based on our workload shift detection solution [7], which describes the DBS workload in terms of *workload models*. We show how these models can be evaluated to identify recurring workloads in a *workload history*. Furthermore, we discuss and evaluate two distinct techniques for the identification of periodicities in a workload history. We also present an approach for predicting future workload shifts from the knowledge about periodicities in the workload. This information is valuable to both DBAs and self-management functions, which may consider it in their reconfiguration decisions. In the future, we also intend to associate concrete DBS configurations with periodic workloads. Thus, it will be possible to set-up a near-optimal DBS configuration without any reconfiguration analysis overhead immediately after a workload change, or even in a pro-active manner.

II. Problem Description

In order to provide a fast adaptation to workload changes, the identification and prediction of periodic workload changes must meet the following requirements:

Recurring Workload Detection – A prerequisite for the detection of periodicities is the identification of recurring workloads. Workloads that are sufficiently similar to workloads in the history must be identified as the same workload.

978-1-4244-6523-1/10/$26.00 © 2010 IEEE

ICDE Workshops 2010

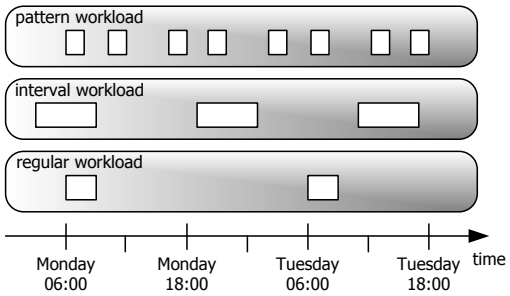


Fig. 1: Types of DBS workload periodicity

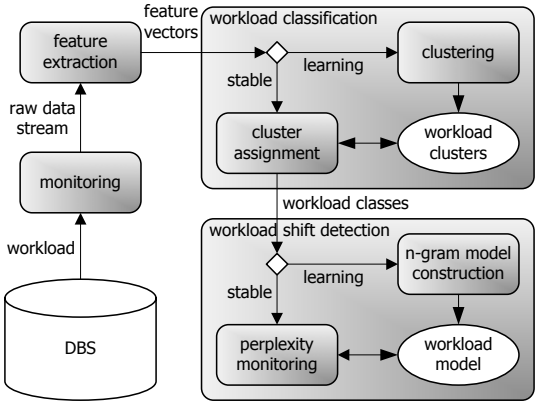


Fig. 2: Workload Monitoring and Analysis Framework

Periodic Pattern Detection – From the workload history all types of workload periodicity must be identified (see Fig. 1): *Regular workloads* appear at specific timestamps, e.g. every morning from 8 AM to 10 AM. *Interval workloads* appear in regular intervals, e.g. every 10 hours. *Pattern workloads* follow a certain activation pattern, e.g. after 2h, 4h, 2h, 4h...

Workload Shift Prediction – From the periodicity knowledge future workload changes must be predicted. As the workload and the periodicities might evolve over time, the predictions must be validated and adapted continuously.

Dependability – Incorrect configurations for a DBS may cause a high processing overhead and business losses. The workload shift predictions must therefore be highly dependable. In particular, it must be possible to define a minimum number of pattern repetitions before periodicity is assumed.

Robustness – In real-world DBS periodic workloads will be subject to fluctuations: First, the starting time and the duration of workloads may vary slightly. Second, there may be exceptions to the usual periodicity, e.g. due to server downtimes. The workload shift identification and prediction must provide robustness to these fluctuations.

Autonomy – Self-Management functions reduce the operation and maintenance costs for DBS by automizing administration tasks. Hence, the workload shift prediction should not impose additional configuration overhead on the DBA again.

As described above, the *Recurring Workload Detection* is the prerequisite for the prediction of periodic workloads. Hence, a precise characterization of observed workloads must be recorded and stored in a workload history. For the characterization of workloads we employ our DBS workload monitoring and analysis framework as described in [7] and [8]. An overview of the existing framework is illustrated in Fig. 2. A *monitoring* component continuously observes the workload from the DBS in terms of the executed queries and updates. The *feature extraction* first extracts a set of relevant features from the statements, e.g. the operation type, the tables accessed, and the grouping requirements (for a detailed discussion see [8]). The resulting feature vectors are then passed to the *workload classification*, where similar statements are grouped. *Clustering* techniques are employed in order to derive these groups in a learning phase. Once the clusters do not change anymore, they are frozen and used for the classification of observed statements (*cluster assignment*). The workload class information is then passed to the *workload shift detection* component, which identifies significant changes in the workload. For this purpose, it creates a n-gram model from the workload classes observed in its learning phase (*n-gram model construction*). This model constitutes a statistical description of the typical workload composition. After the workload model sufficiently well describes the typical workload, the currently observed workload of the DBS is compared to this model by computing the *perplexity* value (for details see [7]). The perplexity is a widely-used measure that describes the quality of an n-gram model for a particualar event source [9]. Informally, it describes how "surprised" the model is by the actually observed events. When the perplexity exceeds a given threshold for some period of time, a workload shift is reported. In this case, the current workload model is discarded and a new model is learned from the changed workload.

Using the existing workload monitoring framework, the major challenges for predicting periodic workloads are: A concept for the comparison of n-gram workload models to identify recurring workloads (Sec. III), a periodicity detection based on the activation timestamps of workload models (Sec. IV), and a prediction of future workload model activations (Sec. V).

III. Identification of Recurring Workloads

To identify recurring workloads, the set of historic workload models must be maintained in a *model pool* (see Fig. 3). Whenever a workload shift occurs and the workload model M_{old} becomes outdated, M_{old} must be compared to the models M_1 to M_n in the pool. If a similar workload has not been observed before, M_{old} is added to the pool with a new model identifier M_{n+1}. Otherwise, M_{old} may be discarded. In either case only the identifier of the model (M_i or M_{n+1}) is required for the subsequent periodicity detection.

For this we pupose we have designed a comparison of n-gram workload models, which integrates seamlessly with the existing workload monitoring concepts: As described in Sec. II, the existing decision on whether or not an observed DBS workload matches the current workload model is based on the *perplexity* computation. This approach can also be generalized for the comparison of two workload models. To

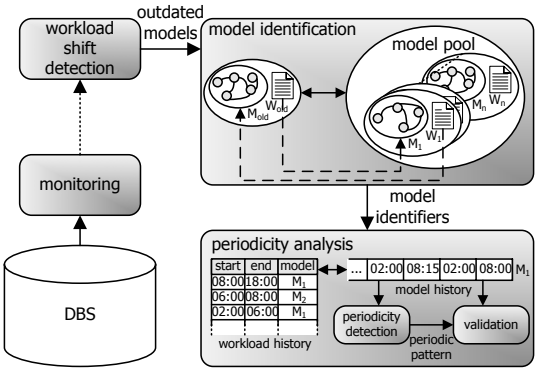
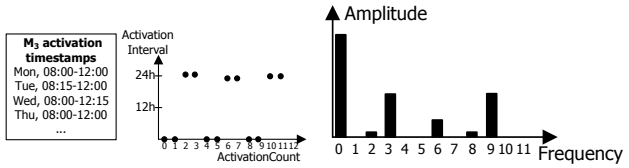


Fig. 3: Workload Identification and Periodicity Analysis



(a) Workload Signal (b) Power Spectrum

Fig. 4: Representation of Model Histories

compare two models M_{old} and M_i, the perplexity of workload W_{old} (the workload that has been used to create M_{old}), is computed on model M_i. If the perplexity value is below the threshold used by the workload shift detection, then the model M_i can be considered to be similar to M_{old}. However, this similarity measure is not symmetric: if the model M_i has many workload classes, but the model M_{old} contains only a small subset, then the perplexity values may be small despite this difference. Hence, also the workload W_i, which has been used to create M_i, must also be compared to model M_{old}. This bi-directional comparison is shown in Fig. 3.

IV. Periodicity Detection

As illustrated in Fig. 3, the workload history comprises the model identifiers along with their begin and end timestamps. To detect periodicities in this type of histories Fourier transforms are typically used (e.g. [10] , [11], [12]). However, the application of this technique to workload models, which is described in Sec. IV-A, does not meet all of the requirements for DBS workload shift prediction. We therfore describe an alternative custom solution in Sec. IV-B.

A. Discrete Fourier Transform

In order to apply the Discrete Fourier Transform (DFT) to a history of DBS workload models, the model history must be converted into a discrete signal. For this purpose, we first separate the models by their identifiers, leading to individual *model histories*. Thus, the workload periodicity detection is simplified to detecting periodicities in the activation timestamps of single a model. Second, each model history is converted into a time-discrete signal. For this purpose we have investigated several approaches, the most appropriate of which was to add a data point to the signal for every activation of the model, where the value of the data point is the duration since its last activation (see Fig. 4a). Two activations are always separated by a data point with the value 0. However, due to the characteristics of the DFT calculation, the transformation results are more accurate if every data point is inserted twice.

With the model history converted to a time-discrete signal, the DFT can be used to compute the power spectrum. If only the fundamental peak and its multiples have amplitudes, then the signal is periodic. However, for DBS workload periodicity detection the robustness requirement demands allowing minor fluctuations. Hence, we define a periodicity measure p, which describes the degree of periodicity of a model M_i as

$$p_{M_i} = \sum_{i=1}^{K} F_{i\omega} / \sum_{i} F_i \tag{1}$$

where ω denotes the fundamental frequency and $F_{i\omega}$ denotes the amplitude of frequency $i\omega$ in the power spectrum. So like the approaches described in [11] and [10], we calculate p_{M_i} as the ratio between the energy in the fundamental frequency and its multiples, and the total energy in the power spectrum. A value of 1 indicates perfect periodicity, and the value drops with increasing non-periodicity. In the example in Fig. 4b, for example, almost all the energy is located at multiples of the fundamental frequency ($i = 3$, $i = 6$, $i = 9$).

The detection of ω, which is a hard problem in general, can be solved in a straight-forward manner in our case: Due to the *dependability*, a minimum number of repetitions r_{min} must have been observed before a workload pattern is confirmed as being periodic. Whenever a model is deactivated, the model history is checked for a pattern that shows exactly r_{min} repetitions. So p_{M_i} is calculated with $\omega = r_{min}$, and if the value is above a certain threshold the model is marked as periodic. It is important to note that this approach requires an immediate check for periodicity for all outdated models M_i that have not yet been marked as periodic.

Using DFT, the p_{M_i} measure provides an elegant solution to the problem of quantifying the periodicity of regular, interval and pattern workloads. However, as will be shown in Sec. VI, the responsiveness of p_{M_i} to non-periodicity depends on the period length. Defining a total threshold for the fluctuations of the model (e.g. 30 minutes) is not possible. Thus, we have designed a second solution named *model interval analysis*.

B. Model Interval Analysis

Like the Fourier approach, this approach considers only the activation timestamps of workload models (the duration is only relevant for the self-management, not for periodicity detection). Furthermore, it also analyses the history of each workload model separately and operates continuously.

The model interval analysis "manually" analyses the intervals between the activations of a workload model M_i, which are stored in an activation interval list I_i. Each activation interval I_{i_j} in this list is computed as the difference between

Algorithm 1: IntervalAnalysis

Input: IntervalList I_i, Min. Periods r_{min} ($r_{min} \geq 2$), Max. Fluctuation $maxFluct$
Output: Periodic pattern P_i

```
1  for k ← 1 to k * r_min > I_i.length do
       /* check range: [0; k * r_min − 1]                    */
2      for i ← 1 to r_min do
3          for j ← 0 to k − 1 do
4              sl[j].add(I_i[((i − 1) * k)) + j]);
5          end
6      end
7      for j ← 0 to k − 1 do
8          if max(sl[j]) − min(sl[j]) > maxFluct then
9              range is aperiodic; continue outer loop;
10     end
11     for j ← 0 to k − 1 do
12         P_i[j] ← avg(sl[j]);
13     end
14     k ← k + 1;
15 end
16 return P_i
```

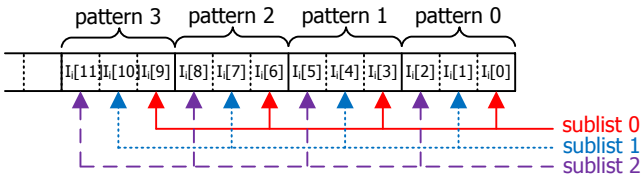


Fig. 5: Illustration of Alg. 1 for $p = 4$ in iteration $k = 3$

two subsequent activations $A_{i_{j-1}}$ and A_{i_j} of M_i. From I_i, the model interval analysis then has to detect the three types of periodic patterns (periodic, interval and pattern workloads). The detection of interval workloads is straight-forward: all intervals in I_i must be identical. For periodic and pattern workloads, a more complex analysisis required: If, for example, a periodic workload is activated every day at 08:00 and 14:00, then this results in the following list of activation intervals: $6h; 18h; 6h; 18h; ...$. Hence, the patterns in I_i may comprise several intervals ($[6h; 18h]$ in this case).

To detect all required patterns, our algorithm (Alg. 1) analyses the activation interval list from the most recent interval $I_i[0]$ to the past. In iteration k, the interval list is analysed from the most recent time $I_i[0]$ to $I_i[k * r_{min} - 1]$. The ratio of this approach is that for $k = 1$ the interval list is analyzed for r_{min} identical subsequent intervals, for $k = 2$ it is analysed for r_{min} pattern occurrences consisting of two intervals, for $k = 3$ for r_{min} pattern occurrences consisting of three intervals, and so on. Fig. 5 illustrates the analysis of an interval list for $k = 3$. The interval values are assigned to k sublists sl, where the j-th interval value of each chunk is assigned to the same sublist $sl[j]$ (**2**-**6**). Thus, the periodicity of the interval values can be judged by comparing the entries within each sublist: if they are identical, then the interval values are periodic with pattern length k. The intervals between the workload model activations are computed as the average of all the observed values in each sublist (**11**-**14**). However, as longer patterns might be found in the interval list, the analysis is continued. Comparing the interval values for equality in the sublists would be too strict because it would not allow for any fluctuations (*robustness*). For this reason the decision on whether or not a given list of interval values is periodic is controlled by a parameter $maxFluct$ (e.g. 30 minutes). Only if the difference of the maximum and minimum interval values of all sublists is smaller than $maxFluct$, the range of interval values is considered to be periodic (**7**-**10**).

The model interval analysis described in Alg. 1 reliably detects the periodic, interval and pattern workloads. In contrast to the Fourier-based approach described in Sec. IV-A, it allows the definition of an absolute value for fluctuations.

V. Shift Prediction and Pattern Validation

The task of the *validation* step in Fig. 3 is to estimate future workload changes and to check whether the identified workload acpatterns are still valid. As in the *periodicity detection* step, we evaluate the appearances of each workload model separately for this purpose. Thus, a workload shift is predicted when one of the average model intervals detected in the periodicity analysis suggests that a model will be activated.

However, on the one hand the intervals in the periodic pattern may evolve over time, or the entire periodic pattern may even become invalid. On the other hand, completely discarding the knowledge about periodic patterns as soon as there is a single exception to the expected appearance is not adequate (*robustness*). The knowledge about the periodicities should instead be adapted over time, and singular events should be considered as being exceptions. For this purpose we have designed Alg. 2, which validates whether the activation of a workload model M_i at timestamp T_{real} conforms to the activation intervals defined in its periodic pattern P_i. It requires the starting time T_{start} of the current period and the counter j of previous model activations in the current period as parameters. From this information, the expected activation time T_{exp} of the workload model is computed by summing up the intervals up to the current activation counter (**1**-**4**). Afterwards, it is checked whether or not the activation has happened within the expected time period (considering $maxFluct$ **5**). If so, then the interval knowledge in $P_i[j]$ is adapted to potential fluctuations by replacing it with the average of the expected interval and the actually observed interval (**6**-**8**). If the activation is premature, then it is considered as an occasional singular event and is ignored (**15**). If the activation is late, a (persistent) failure counter $missing$ is increased. If this counter exceeds a given threshold $maxFail$, the periodic pattern is discarded and a new pattern must be identified using the concepts described in Sec. IV (**11**-**12**). Otherwise the workload model activation counter j is increased by 1, and the validation function is called recursively (**14**).

Algorithm 2: ActivationValidation

Input: Pattern P_i, Period Start T_{start}, Activation No. j $(0 \leq j < P_i.length)$, Activation T_{real}, Max. Fluctuation $maxFluct$, Max. Failures $maxFail$
Output: true/false

```
1  T_exp ← T_start;
2  for i ← 0 to j do
3      T_exp ← T_exp + P_i[i];
4  end
5  if |T_exp − T_real| < maxFluct then
6      I_obs ← T_real − (T_exp − P_i[j]);
7      P_i[j].set((P_i[j] + I_obs)/2);
8      return true;
9  else
10     if T_real > T_exp then
11         if ++missed > maxFail then
12             return false;
13         else
14             return ActivationValidation(P_i, T_start, j + 1,
               T_real, maxFluct, maxFail);
15     return true;
```

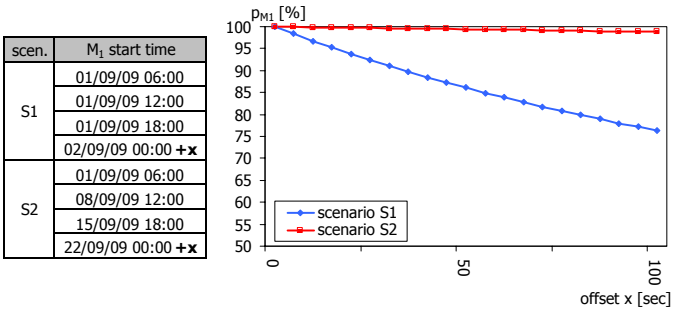


Fig. 6: Effects of fluctuations on p_{M_1}

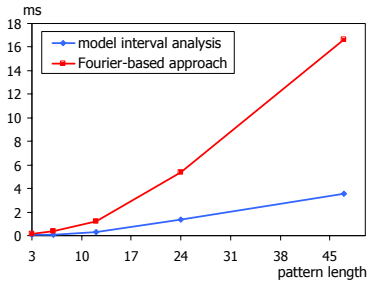


Fig. 7: Periodicity Detection Overhead Analysis

VI. Evaluation

For the evaluation of our periodicity detection approach we have performed a functional and an overhead analysis. The functional analysis has evaluated the reaction of the periodicity measure p of the Fourier approach described in Sec. IV-A to fluctuations in the patterns. For this purpose we have defined the workload scenario *S1* described in Fig. 6: Starting Sept. 1st, the workload model M_1 in this scenario appears in regular intervals of six hours (the duration of the model is not relevant for the periodicity measure value). In order to evaluate the responsiveness of the periodicity measure to fluctuations in this model history, we have iteratively increased of start timestamp of the fourth appearance of the model by an offset x. For each offset x we have then re-calculated the periodicity measure p_{M_i} for $r_{min} = 3$. As illustrated in Fig. 6, the periodicity measure in this case constantly drops from a value of 1 at the offset 0 to below 0.8 for the offset 100.

In scenario *S2* we have then increased the activation intervals of the model M_1 to one week and again calculated p_{M_i} for different offset values of the fourth activation. Fig. 6 shows that in this scenario p_{M_i} decreases far slower than in $S1$. So the value p_{M_i} is relative to the period length of workload pattern. Hence, the Fourier-based approach only supports the definition of fluctuation thresholds that are relative to the period length. While this may be the desired behaviour in some cases, it does not allow the definition of a fixed time limit for acceptable fluctuations. If this functionality is required, our model interval analysis approach can be used instead. For this approach, we have validated the reliable detection of the regular workloads, interval workloads and pattern workloads using a variety of sample workload histories.

To assess the overhead caused by the Fourier-based approach and by the model interval analysis, we have applied both of them to a sample workload history (length 250). The workload histories had to be analysed for five repetitions ($r_{min} = 5$) of a workload pattern of increasing length. Fig. 7 illustrates the computation times of both approaches. It can be seen that although the overhead of both approaches increases with an increasing pattern length, the model interval analysis periodicity detection is always more efficient than the Fourier-based approach using the FFT algorithm [13]. Still, the absolute values of the periodicity are in both cases very small (less than 10 ms for a pattern with 40 model activations).

The overhead for the identification of recurring workloads is determined by the effort for the perplexity computation. Our evaluation in [7] has shown that this value can be efficiently calculated: for a probe of 100 workload events and a n-Gram length of 3, the computation took less than 5ms on an average (independent of the model size). Due to the bi-directional comparison of workload models (cf. Sec. III), the comparison of an outdated model M_{old} with a model from the pool would therefore take less than 10 ms for this probe size. Considering that this overhead would be caused only a few times a day (only when there is a workload change), we consider this overhead to be acceptable.

VII. Related Work

The approach described in [14] predicts changes between decision support (DSS) and OLTP workloads of a DBS. As in our approach, analytical methods are applied to historic workload information in order to estimate the time interval for shifts. However, [14] does not build a model of the SQL

workload, but maintains a history of "DSSness"-indicators, which describe the similarity of the workload to a typical DSS workload. If the consolidated "DSSness"-indicators exceed or fall below a certain threshold, a shift is predicted. While this approach is reasonable for distinguishing two clearly characterized workload types, it cannot be applied for multiple, previously unknown workloads. Furthermore, the period length has to be defined in advance (e.g. one day), whereas it is automatically detected in our approach.

[11] identifies periodicities in the number of SQL statements submitted to the DBS. For this purpose it also identifies the fundamental peaks and harmonics in the Fourier transforms and calculates a periodicity metric on this information. Unlike our approach, [11] does not allow the identification of periodicities between different *types* of workloads. Instead, only periodic structures in the amount of workload submitted to the DBS can be identified, which does not allow any conclusion about the appropriate DBS configuration. Furthermore the approach described in [11] does not support the definition of a minimum number of pattern repetitions, or an absolute limit for the length of the acceptable fluctuations.

Today's commercial DBMS also offer the storage of historic workload information, e.g. the AWR [15] in Oracle or the Workload Manager and Event Monitor [16] in DB2. Although the statistical and usage information of these repositories is exploited for various self-management tasks, it is not yet analyzed for recurring workload situations or periodicity.

Workload characterizations based on n-Gram models are also used in [17]. This work is directed at the inference of user sessions from SQL statement traces. The information about the user sessions is intended to be used for the prediction of queries, e.g. for cache replacement strategy optimization. Hence, the prediction capabilities are limited to individual statements within a transaction.

Following the goal of server capacity management, [12] detects periodic patterns in CPU or memory demands of enterprise applications. The pattern analysis is performed using Fourier transforms and autoregression. The degree of periodicity is then judged by a combination of the deviation in the time domain and in the value domain. Although the described periodicity detection are in some parts similar, there are some major differences: [12] assumes that the workload information is a continuous signal, which is not the case for the discrete activation of DBS workload models. Furthermore, the acceptance of fluctuations in the value domain are not suitable for our periodicity detection scenario, and there are no provisions to provide strict limits for the acceptable fluctuations.

The prediction of the workload in a mainframe operating system is discussed in [18]. In contrast to our approach, the authors do not analyse the workload for periodicity explicitly, but train neural networks from historic workloads. The results show that this method is applicable for short-term predictions only. [19] presents a technique for predicting the workload in terms of CPU utilization in grids. Like [18], this approach does not focus on identifying long-term periodicity, but predicts the expected short-term grid utilization using time series analysis.

VIII. Conclusions

In this paper we have proposed two approaches for the identification of periodic workload changes. While the threshold of acceptable fluctuations in the Fourier approach depends on the period length, our custom periodicity detection approach allows the definition of strict time limits. Furthermore, we have proposed an algorithm that adapts the periodicity knowledge to workload evolutions. In the future, we plan to use this information in order to associate DBS configurations with particular workload models. Thus, we intend to predict changes in the workload, to validate that the new workload actually matches predicted workload and implement the associated DBS configuration without having to perform an expensive reconfiguration analysis.

References

[1] B. Dageville *et al.*, "Automatic SQL Tuning in Oracle 10g," in *Proc. of the* 30^{th} *Intl. Conf. on Very Large Data Bases*, M. A. Nascimento *et al.*, Eds. Morgan Kaufmann, 2004, pp. 1098–1109.

[2] D. C. Zilio *et al.*, "Recommending Materialized Views and Indexes with IBM DB2 Design Advisor," in *Proc. of the* 1^{st} *Int. Conf. on Autonomic Computing*. IEEE, 2004, pp. 180–188.

[3] E. Kwan *et al.*, "Automatic Configuration for IBM DB2 Universal Database," in *10. Konf. fr DBS für Business, Technologie und Web*. Gesellschaft für Informatik, 2003, pp. 620–629.

[4] B. Dageville and M. Zaït, "SQL Memory Management in Oracle9i," in *Proc. of* 28^{th} *Intl. Conf. on Very Large Data Bases*, P. A. Bernstein *et al.*, Eds. Morgan Kaufmann, 2002, pp. 962–973.

[5] A. J. Storm *et al.*, "Adaptive Self-Tuning Memory in DB2," in *Proc. of the* 32^{nd} *Int. Conf. on Very Large Data Bases*. ACM, 2006, pp. 1081–1092.

[6] N. Bruno and S. Chaudhuri, "An Online Approach to Physical Design Tuning," in *Proc. of the* 23^{rd} *Int. Conf. on Data Eng.* IEEE, 2007, pp. 826–835.

[7] M. Holze and N. Ritter, "Autonomic Databases: Detection of Workload Shifts with n-Gram-Models," in *Proc. of the* 12^{th} *East Europ. Conf. on Adv. in Databases and Inf. Syst.* Springer, 2008, pp. 127–142.

[8] M. Holze *et al.*, "Consistent On-Line Classification of DBS Workload Events," in *Proc. of the* 18^{th} *Int. Conf. on Information and Knowledge Management*. ACM, 2009, pp. 1641–1644, to appear.

[9] G. Fink, *Markov Models for Pattern Recognition*, 1st ed. Springer, 2007.

[10] R. Polana and R. Nelson, "Detecting Activities," in *Proc. of the Intl. Conf. on Comp. Vision and Pattern Recogn.* IEEE, 1993, pp. 2–7.

[11] A. D. Buckler, "Workload Periodicity Analyzer for Autonomic Database Components," U.S. Patent US 7,509,336 B2, 03 24, 2009.

[12] D. Gmach *et al.*, "Workload analysis and demand prediction of enterprise data center applications," in *Proc. of the* 10^{th} *IEEE Workload Characterization Symposium*. IEEE, 2007, pp. 171–180.

[13] J. Cooley and J. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[14] S. S. Elnaffar and P. Martin, "An Intelligent Framework for Predicting Shifts in the Workloads of Autonomic Database Management Systems," in *Proc. of Intl. Conf. on Advances in Intelligent Systems*. IEEE, 2004.

[15] K. Dias *et al.*, "Automatic Performance Diagnosis and Tuning in Oracle," in *Proc. of the* 2^{nd} *Biennial Conf. on Innovative Data Systems Research*. Online Proceedings, 2005, pp. 84–94.

[16] W.-J. Chen *et al.*, *DB2 Workload Manager for Linux, UNIX, and Windows*, 1st ed., IBM, 2008, Redbook.

[17] Q. Yao *et al.*, "Finding and analyzing database user sessions," in *Proc. of the* 10^{th} *Intl. Conf. on Database Systems for Advanced Applications*, ser. LNCS, L. Zhou *et al.*, Eds., vol. 3453. Springer, 2005, pp. 851–862.

[18] M. Bensch *et al.*, "Self-learning prediction system for optimisation of workload management in a mainframe operating system," in *Proc. of the* 9^{th} *Intl. Conf. on Enterprise Information Systems*, J. Cardoso *et al.*, Eds., vol. AIDSS, 2007, pp. 212–218.

[19] Y. Wu *et al.*, "Adaptive Workload Prediction of Grid Performance in Confidence Windows," *IEEE Trans. Parallel and Distrib. Syst.*, preprint.