Discovering Periodic Patterns in System Logs*

Marcin Zimniak[1], Janusz R. Getta[2], and Wolfgang Benn[1]

[1] Faculty of Computer Science, TU Chemnitz, Germany
{marcin.zimniak,benn}@cs.tu-chemnitz.de
[2] School of Computer Science and Software Engineering,
University of Wollongong, Australia
jrg@uow.edu.au

Abstract. Historical information stored in the software system logs, audit trails, traces of user applications, etc. can be analysed to discover the patterns in periodic variations of levels and structures of the past workloads. These patterns allow for the estimation of intensities and structures of the future workloads. The correctly anticipated future workloads are used to improve performance of software systems through appropriate allocation of computing resources and through restructuring of associated system support. This work defines a concept of *periodic pattern* and presents the algorithms that find the periodic patterns in the traces of elementary and complex operations on data recorded in the system logs.

1 Introduction

The typical approaches to performance tuning of software systems either find the software components that have significant impact on performance or find a group of software components whose simultaneous processing contributes to the performance bottlenecks [1]. Optimization of software components restructures associated system support. It relocates data containers to the faster storage devices, runs processes on the faster processors, adds more resources and computational power, increases the priorities of performance critical processes, etc. Optimization through elimination of bottlenecks restructures the functionality of software components involved in the collisions during their simultaneous processing.

Automated performance tuning of software systems [2] implements an "observer" module that automatically changes a level of system support or resolves the collisions whenever it is necessary. An important factor is the ability of an "observer" to anticipate the low workload times when re-balancing of system support or elimination of bottlenecks can be done. An important question is

* *Copyright* © 2014 *by the paper's authors. Copying permitted only for private and academic purposes.* In: T. Seidl, M. Hassani, C. Beecks (Eds.): Proceedings of the LWA 2014 Workshops: KDML, IR, FGWM, Aachen, Germany, 8-10 September 2014, published at http://ceur-ws.org

whether it is possible to predict the future characteristics of workloads from information about the past behaviour of a software system. A promising approach is to utilize the periodic repetitions of data processing caused by the "repetitive nature" of real world processes. For example, students enrolling courses at the beginning of each session, accountants processing financial data at the end of financial year, footballers playing games every week, etc. trigger the repetitions of the same data processing cycles. An objective of this work is to provide efficient algorithms for discovery of periodic patterns in the traces of elementary and complex operations on data recorded in the system logs. We assume an abstract model of computations where a system log is represented by a sequence of nested n-ary operations on data later on called a *workload trace*. A workload trace is sequence of groups of operations computed in the same periods of time. A workload trace is "reduced" by elimination of all nested operations that provide the arguments for only one higher level operation. Then, the processing patterns of "children" operations are the same as the patterns of "parent" operations.

The paper is organized in the following way. The next section refers to the previous works in the related research areas. Section 3 defines the basic concepts and a model of workload trace. A concept of periodic pattern in a system workload is defined in Section 4 and discovering of elementary periodic patterns is explained in Section 5. Section 6 concludes the paper.

2 Related work

Data mining techniques that inspired the works on periodic patterns came from the works on mining association rules [3] and later on from mining frequent episodes [4] and its extensions on mining complex events.

The problem seems to be very similar to a typical periodicity mining in time series [5], where analysis is performed on the long sequences of elementary data items discretized into a number of ranges and associated with the timestamps. In our case, the input data is a sequence of complex data processing statements, e.g. SQL statements and due to its internal structure cannot be treated in the same way as analysis of elementary data elements in time series or genetic sequences.

The recent approaches, which addressed full periodicity, partial periodicity, perfect and imperfect periodicity [6], and asynchronous periodicity [7] are all based on fixed size and adjacent time units and fixed length of discovered patterns.

Our problem is also similar to a problem of mining cyclic association rules [8] where an objective is to find the periodic executions of the largest sets of items that have enough support.

Invocation of operation on data along the various points in time can be easily described by temporal predicates within a formal scope of temporal deductive database systems [9]. The reviews of data mining techniques based on analysis of ordered set of operations on data performed by the user applications are available in [10], [11]. The model of periodicity considered in this paper is consistent with the model proposed in [12].

3 Basic concepts

The operations processed by the *software components* $c_1, \ldots, c_n$ are recorded in an *operation trace*. A *trace* of a software component c_i is a finite sequence of pairs $<p_{i_1}{:}t_{i_1}, \ldots, p_{i_n}{:}t_{i_n}>$ where each t_{i_j} is a timestamp when an operation p_{i_j} has been processed.

A *system trace* A is a sequence of interleaved trails of software components processed in a certain period of time. For example, a sequence $<p_{i_1}{:}t_{i_1}, p_{j_1}{:}t_{j_1}, p_{i_2}{:}t_{i_2}, p_{j_2}{:}t_{j_2}>$ is a sample system trace from the processing of software components c_i, and c_j.

Let $<t_{start}, t_{end}>$ be a period of time over which a system trace is recorded. A *time unit* is a pair $<t, \tau>$ where t is a start point of a unit and τ is a length of the unit. A nonempty sequence U of n disjoint time units $<t^{(i)}, \tau^{(i)}>$ $i = 1, \ldots, n$ over $<t_{start}, t_{end}>$ is any sequence of time units that satisfies the following properties: $t_{start} \leq t^{(1)}$ and $t^{(i)} + \tau^{(i)} \leq t^{(i+1)}$ and $t^{(n)} + \tau^{(n)} \leq t_{end}$.

A multiset M is defined as a pair $<S, f>$ where S is a set of values and $f : S \rightarrow N^+$ is a function that determines multiplicity of each element in S and N^+ is a set of positive integers. In the rest of this paper we shall denote a multiset $<\{T_1, \ldots, T_m\}, f>$ where $f(T_i) = k_i$ for $i = 1, \ldots, m$ as $(T_1^{k_1} \ldots T_m^{k_m})$. We shall denote an empty multiset $<\emptyset, f>$ as $\emptyset$ and we shall abbreviate a multiset (T^k) to T^k and T^1 to T.

A *workload trace* of an operation T is a sequence W_T of $|U|$ multisets of operations such that $W_T[i] =<\{T\}, f_i>$ and $f_i(T) = |T.timestamp(i)|$ $\forall i = 1, \ldots, |U|$ i.e. $f_i(T)$ is equal to the total number of times an operation T was processed in the i-th time unit $U[i]$.

Let $\mathbf{T}$ be a set of all operations obtained from a system trace A. A *workload trace* of A is denoted by W_A and $W_A[i] = \biguplus_{T \in \mathbf{T}} W_T[i], \forall i = 1, \ldots, |U|$, i.e. it is a sum of workload traces of all operations processed in U.

4 Periodic patterns

A *periodic pattern* is a tuple $<\mathcal{T}, U, b, p, e>$ where $\mathcal{T}$ is a nonempty sequence of multisets of operations, U is a sequence of disjoint time units, $b \geq 1$ is a number of time unit in U where the repetitions of $\mathcal{T}$ start, $p \geq 1$ is the total number of time units after which processing of $\mathcal{T}$ is repeated in every processing cycle, $e > b$ is a number of time unit in U where the processing of $\mathcal{T}$ is performed for the last time. A sequence of multisets $\mathcal{T}$ may contain one or more empty multisets. The positional parameters b, p, and e of a periodic pattern must satisfy a property $(e - b) mod\ p = 0$. A value $c = \frac{e-b}{p} + 1$ is called as the *total number of cycles* in the periodic pattern.

Let $\mathcal{T}_{ext}$ be a sequence of multisets of operations obtained from $\mathcal{T}$ and extended on the right with $e - b$ empty multisets. Then, a *workload trace of a periodic pattern* $<\mathcal{T}, U, b, p, e>$ with the total number of cycles $c = \frac{e-b}{p} + 1$ is a sequence $W_{\mathcal{T}}$ of $e - b + |\mathcal{T}|$ multisets of operations such that $W_{\mathcal{T}}[i] =$

$\mathcal{T}_{ext}[g(i)] \uplus \mathcal{T}_{ext}[g(i-p)] \uplus \mathcal{T}_{ext}[g(i-2*p)] \uplus \ldots \mathcal{T}_{ext}[g(i-(c-1)*p)]$ for $i = 1, \ldots, e-b+|\mathcal{T}|$ where a function g is computed such that `if` $x > 0$ `then` $g(x) = x$ `else` $g(x) = x + e - b + |\mathcal{T}|$.

For example, $<\emptyset TV, U, 1, 1, 3>$ is a periodic pattern where processing of a sequence of operations $\emptyset TV$ starts in the time units 1, 2, and 3 and its workload trace is a sequence of multisets $\emptyset T(VT)(VT)V$. The periodic pattern has 3 cycles.

Let $|W_{\mathcal{T}}|$ be the total number of elements in $W_{\mathcal{T}}$. Let v be the total number of elements in $W_{\mathcal{T}}$ such that $W_{\mathcal{T}}[i] \sqsubseteq W_A[b+i-1]$ for $i = 1, \ldots, e-b+|\mathcal{T}|$. Then, we say that a periodic pattern $<\mathcal{T}, U, b, p, e>$ is *valid in a system trace A with a support* $0 < \sigma \leq 1$ if $W_{\mathcal{T}}[1] \sqsubseteq W_A[b]$ and $W_{\mathcal{T}}[e-b+|\mathcal{T}|] \sqsubseteq W_A[e+|\mathcal{T}|]$ and $\sigma \leq v/|W_{\mathcal{T}}|$.

For example, a periodic pattern $<(T^2V)\emptyset W, U, 2, 3, 8>$ has a workload trace $(T^2V)\emptyset W(T^2V)\emptyset W(T^2V)\emptyset W$. The pattern is valid in a system trace A with support $\sigma = 1$ if every element of its workload trace is included in a workload trace W_A from position 2 to position 10.

A periodic pattern $<\mathcal{T}, U, b, p, e>$ such that $\mathcal{T} = T^{i_1} \ldots T^{i_n}$ where $i_k, n \geq 1$ and $p \geq |\mathcal{T}|$ is called as a *homogeneous periodic pattern*. For example, a periodic pattern $<T^7T^2T^{11}, U, 10, 4, 22>$ is a homogeneous periodic pattern, which has four cycles.

5 Discovering periodic patterns

A method for discovering periodic patterns in database audit trails proposed in [12] iterates over the dimensions of syntax trees of SQL statements retrieved from an audit trail and the dimensions of positional parameters b, p, and e. The algorithm finds only homogeneous periodic patterns such that $\mathcal{T} = T^k$ and its computational complexity is approximately $O(k * n^3)$ where $0 < k < 1/8$ and n is the total number of time units of an audit trail.

An approach to mining periodic patterns proposed in this paper is based on two algorithms. The first algorithm finds in a workload trace W_A the longest subsequence W_T of nonempty multisets, which is included in the largest number of times in the trace W_A. Then, W_T is passed to the second algorithm to generate the candidate periodic patterns and to return the candidate pattern with the highest support to the first algorithm. Next, the first algorithm saves the periodic pattern, it removes from W_A a workload trace of the pattern, and it repeats itself until it is possible to find a new W_T which has at least two separate subsequences.

Algorithm 1
Let $\mathbf{T}$ be a set of all operations included in a workload trace W_A. The following algorithm iteratively performs the following steps over the operations $T \in \mathbf{T}$. At the beginning a set of periodic patterns $\mathcal{P}$ is empty.

(1) We transform W_T into a sequence of numbers (words with length equal to $|U|$), $< f_i(T) >$, such that $f_i(T) \in N_0, \forall i = 1, \ldots, |U|$, $W'_T := W_T$.

(2) If there are no empty multisets in W'_T then we transform W'_T as follows, $W'_T := W'_T -$ (workload trace of $< T, 1, 1, |U| >$) the smallest number of $k_{i_{min}}$ times such that in the result of transformation we obtain at least one empty multiset, i.e. there is at least one zero in a sequence (word) $< f'_i(T) >$. We add a periodic pattern to a set $\mathcal{P}$, $\mathcal{P} := \mathcal{P} \cup < T^{k_{i_{min}}}, 1, 1, |U| >$.

(3) We find in W'_T all longest sub-sequences $\{W_i(W'_T)\}$ in a sense of inclusion of multisets over their components and such that there exists the largest number of the same sub-sequences whose length is equal to $\max || < f'_{i_j}(T), ..., f'_{i_k}(T) > ||$.

(4) For each individual $W_i(W'_T)$ we apply **Algorithm 2** described below and from all periodic patterns found we pick a pattern with the largest value of support σ.

(5) If from a step (4) we get pp_{hom} then $\mathcal{P} := \mathcal{P} \cup pp_{hom}$ and $W'_T := W'_T -$ (workload trace of pp_{hom}) and we return to step (3). If a step (4) returns no solutions then we progress to the next step.

(6) It is possible to remove from any valid periodic pattern any leading or trailing sequence of T and still get a periodic pattern valid in the same workload trace. We search W'_T for shorter homogeneous periodic patterns $pp_{hom}{}'$. We insert each $pp_{hom}{}'$ found into $\mathcal{P}$ and $W'_T := W'_T -$ (workload trace of $pp_{hom}{}'$). We return to step (3).

Algorithm 2

An input to the second algorithm is a workload trace W_T, a given sequence of multisets of operations $\mathcal{T}$, the locations of the first (f) and the last (t) instances of $\mathcal{T}$ in W_T. The parameters of a candidate periodic pattern in W_T must satisfy the following linear Diophantine equation:

$$c * |\mathcal{T}| + (c-1) * d = e - b + |\mathcal{T}| \tag{1}$$

where d is a distance between the instances of $\mathcal{T}$ in the pattern and c is the total number of cycles in the pattern. To solve the equation we assume that $b = f$, $e = t$. Let d_{min} (d_{max}) be the shortest (the longest) distance between any two locations where $\mathcal{T}$ is included in W_T. The algorithm consists of the following steps.

(1) We make a set of candidate periodic patterns P empty.

(2) We iterate over the values of $d = d_{min}, d_{min} + 1, \ldots, d_{max}$.

(2.1) For a given value of d we find the following values of c and r:

$$c = \frac{e-b}{|\mathcal{T}| + d} + 1 \tag{2}$$

$$r = (e-b) \ mod \ (|\mathcal{T}| + d) \tag{3}$$

(2.2) Let $p = |\mathcal{T}| + d$. We create the periodic patterns $<\mathcal{T}, b, p, b + p * (c-1)>$, $<\mathcal{T}, b+1, p, b + p * (c-1) + 1>$, $\ldots$, $<\mathcal{T}, b + r, p, b + p * (c-1) + r>$. We append the periodic patterns found to a set of candidate periodic patterns P. If available we pick the next value of d and we return to step (2.1).

(3) A set of candidate homogeneous periodic patterns P is returned to the first algorithm.

6 Summary and conclusions

Discovering the complex periodic patterns in the system logs is a difficult and time consuming task. This work defines a concept of *periodic pattern* and shows how to find the periodic patterns in the the system logs. An approach described here shows that it is easier to find the simple periodic patterns and later on to compose them into the complex ones instead of directly searching for all complex patterns. The discovered periodic patterns can be used to model future workload after the old applications are replaced with the new ones or the new applications are added to a system. It is also easier to reconcile the new audit trails with the collections of periodic patterns discovered from the previous system logs than to integrate the complete logs.

References

1. Osterhage, W.: Computer Performance Optimization. Springer-Verlag (2013)
2. Bruno, N., ed.: Automated Physical Database Design and Tuning. CRC Press Taylor and Francis Group (2011)
3. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of The 1993 ACM SIGMOD Intl. Conf. on Management of Data. (1993) 207–216
4. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery **1** (1997) 259–289
5. Rasheeed, F., Alshalalfa, M., Alhajj, R.: Efficient periodicity mining in time series databases using suffix trees. IEEE Transactions on Knowledge and Data Engineering **23**(1) (2011) 79–94
6. Huang, K.Y., Chang, C.H.: SMCA: A general model for mining asynchronous periodic patterns in temporal databases. IEEE Transactions on Knowledge and Data Engineering **17**(6) (2005) 774–785
7. Yang, J., Wang, W., Yu, P.S.: Mining asynchronous periodic patterns in time series data. IEEE Trans. on Knowl. and Data Eng. **15**(3) (March 2003) 613–628
8. Özden, B., Ramaswamy, S., Silberschatz, A.: Cyclic association rules. In: Proceedings of the Fourteenth International Conference on Data Engineering. (1998) 412–421
9. Baudinet, M., Chomicki, J., Wolper, P.: Temporal deductive databases (1992)
10. Laxman, S., Sastry, P.S.: A survey of temporal data mining. Sadhana, Academy Proceedings in Engineering Sciences **31**(2) (2006) 173–198
11. Roddick, J.F., Society, I.C., Spiliopoulou, M., Society, I.C.: A survey of temporal knowledge discovery paradigms and methods. IEEE Transactions on Knowledge and Data Engineering **14** (2002) 750–767
12. Zimniak, M., Getta, J., Benn, W.: Deriving composite periodic patterns from database audit trails. In: The 6th Asian Conference on Intelligent Information and Database Systems. (2014) 310–321