A Multiple Query Optimization Scheme for Change Point Detection on Stream Processing System

[Position Paper]

Masahiro Oke and Hideyuki Kawashima(✉)

University of Tsukuba, 1-1-1, Tennodai, Tsukuba, Japan
oke@kde.cs.tsukuba.ac.jp, kawasima@cs.tsukuba.ac.jp

Abstract. To accelerate simultaneous execution of multiple change point detection (CPD) queries, this paper proposes to apply the multiple query optimization scheme which has been studied in DBMS or DSMS. We propose to share a part of steps of CPD procedures, and we propose an algorithm for the sharing. The result of experiments showed that our proposal reduces more than 80 % internal steps and achieved 5 times performance improvement. To the best of our knowledge, this is the first work that applies the multiple query optimization scheme for CPD.

Keywords: Change point detection · Multiple query optimization

1 Introduction

A variety of techniques are used to detect malware. Data stream management systems are effective tools for the problem. For example, TCP syn flood detection is realized group-by-aggregate continuous query as shown in [5]. It should be noted that new types of malware appear every day, and [1] people do not accept to be infected by malware. Therefore new malware detection techniques will be developed continually beyond relational operators. To detect malware, a signature based detection method is used as a basic method. This method does not perform if a signature is not yet created when malware arrives.

Machine learning techniques or data mining techniques are sometimes adopted for such malware. For example, an incident analysis system NICTER [1] uses change point detection technique (CPD) [2] to detect malware in real-time using its dark net traffic generated by more than 160,000 unused IP addresses. CPD is an outlier detection technique for time series data based on autoregressive model with the concept of discounting. CPD requires 6 parameters to be executed. The parameters are deeply related to detection accuracy as other machine learning techniques. Choosing a single parameter set that shows the best accuracy is desirable. Such a choice is impossible without predicting the future. Therefore, multiple parameter sets should be chosen and

M. Castellanos et al. (Eds.): BIRTE 2013 and 2014, LNBIP 206, pp. 150–158, 2015.
DOI: 10.1007/978-3-662-46839-5_10

multiple CPD should run simultaneously to improve the accuracy of malware detection. We are developing Falcon, yet another DSMS which provides not only relational operators but also data mining operators including CPD.

On running Falcon, we found a serious performance issue when running N processes. To accelerate simultaneous execution of multiple CPD procedures, this paper proposes a multiple query optimization scheme for CPD. For relational database system, a variety of multiple query optimization (MQO) techniques have been studied such as sharing common sub-expressions [3]. On the other hand, our proposal is dedicated for CPD, and it shares internal steps in CPD, which is different from usual MQO. To the best of our knowledge, this is the first work that describes MQO for CPD.

The rest of this paper is organized as follows. Section 2 describes CPD in detail. Section 3 proposes an efficient computation scheme for multiple CPD operations. Section 4 evaluates our proposal techniques. Finally Sect. 5 concludes this paper.

2 Related Work: Change Point Detection

2.1 CPD (Change Point Detection)

Change point detection is an outlier detection technique for time series data proposed by Takeuchi and Yamanishi [2]. CPD can detect points that change dramatically on time series data. An incident analysis system NICTER [1] uses CPD to detect occurrences of malware over network packets arrived at its dark net consisting of more than 160,000 nodes. CPD uses an auto-regressive (AR) model as a time series model. Roughly saying, CPD is constituted of 4 steps as shown in Fig. 1. They are (step 1 and 3) learning probability density and (step 2 and 4) scoring and moving averages of scores. The learning algorithms used in step 1 and 3 are referred to as sequential discounting AR model learning (SDAR). SDAR can deal with non-stationary time series data. Traditional learning algorithms on AR model are batch processing, while SDAR is an online learning algorithm that executes learning processes with the arrival of new data. Therefore its learning cost is less than batch processing scheme.

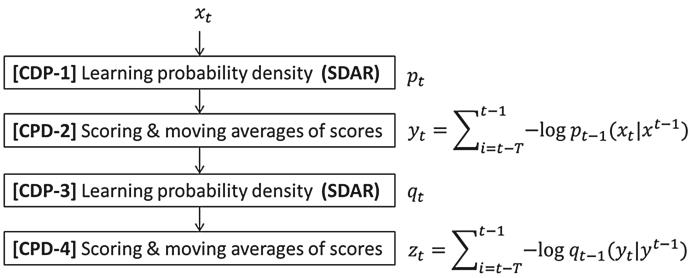

Fig. 1. Steps for CPD

2.2 SDAR (Sequentially Discounting Auto Regressive)

The SDAR algorithm shown in step 1 and 3 in Fig. 1 is a learning algorithm of AR model with discounting and online learning features. The discounting feature decreases the effect of a t-step previous value. The effect is modified to $(1 - R) \times t$, however the range of R is between 0 to 1. Input parameters of SDAR algorithm are R (discounting ratio), $\hat{\mu}$(mean), C_j(auto covariance), $\hat{\omega}_j$ and $\hat{\Sigma}$(covariance). All of the five parameters should be provided by a user before starting the SDAR algorithm. We explain five steps of SDAR shown in Fig. 2.

First, on reading a new data object x_t, SDAR executes the following computations shown as [SDAR-1] and [SDAR-2] in Fig. 2.

$$\hat{\mu} = (1 - R)\hat{\mu} + Rx_t$$

$$C_j = (1 - R)C_j + R(x_t - \hat{\mu})(x_{t-j} - \hat{\mu})^T$$

Then we write a Yule-Walker equation as follows.

$$\sum_{t-1}^{K} \omega_i C_{j-i} = C_j$$

By solving the equation, we obtain $\omega_1 \ldots \omega_K$. This is [SDAR-3] in Fig. 2. Using them, we obtain $\hat{x}_t$ and $\hat{\Sigma}$ as follows. They are [SDAR-4] and [SDAR-5] in Fig. 2.

$$\hat{x}_t = \sum_{i=1}^{K} \hat{\omega}_i(x_{t-i} - \hat{\mu}) + \hat{\mu}$$

$$\hat{\Sigma} = (1 - R)\hat{\Sigma} + R(x_t - \hat{x}_t)(x_t - \hat{x}_t)^T$$

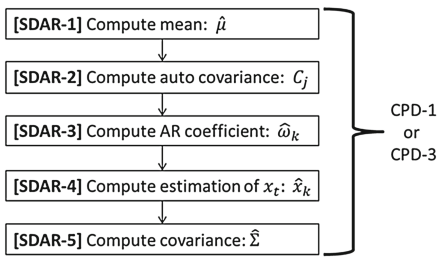

Fig. 2. Steps of SDAR

3 Multiple Query Optimization for CPD

3.1 Problem on CPD: Influence of Parameters to Detection Accuracy

CPD has 3 kinds of parameters for input: discounting parameter R, AR model order K and moving time T. CPD requires 6 parameters: α_R, α_K, α_T, β_R, β_K and β_T, which will be explained in Sect. 3.2. Parameters α_R, α_K and α_T are for 1st stage learning which

is shown as CPD-1 and CPD-2 in Fig. 1. On the other hand, β_R, β_K and β_T are for 2nd stage learning which is shown as CPD-3 and CPD-4 in Fig. 2. It should be noted that CPD-1 and CPD-3 are the same, and CPD-2 and CPD-4 are the same, which is the reason why CPD is referred to as 2 step stage learning. Figure 3 shows the effect of parameter sets to detection accuracy for CPD. In Fig. 3, horizontal axis shows time. Left vertical axis shows frequency (the number of accesses) shown by blue graph which locates on the upper side and changes frequently. Right vertical axis shows CPD score shown by brown graph which locates on the lower side and changes infrequently. In the left one, CPD score is responded by outlier accesses. On the other hand, in the right one, CPD score does not show any response with regard to frequency. Left parameter set is appropriate while right parameter set is inappropriate.

If our motivating applications are for stored database, then we can tune parameters and can find out the best parameter set. Streaming applications, however, does not take such a chance. Therefore, to reduce false negatives, multiple parameter sets should be tried simultaneously. Discounting parameter R decreases influence of past data. Therefore if R is small, CPD score is greatly affected by past data. AR model order K expresses the number of data used for learning. If we set K large, an AR model learns from more data. However, it should be noted that larger K does not directly mean better model in the viewpoint of Akaike Information Criterion. Moving time T is used to make outlier scores smoothly and to compute change point score. If T is large, then change detection is for long duration. If T is small, then it is for short duration. Parameters should be set appropriately for each malware. Since malware can be unknown, many types of parameter sets should be carefully tuned.

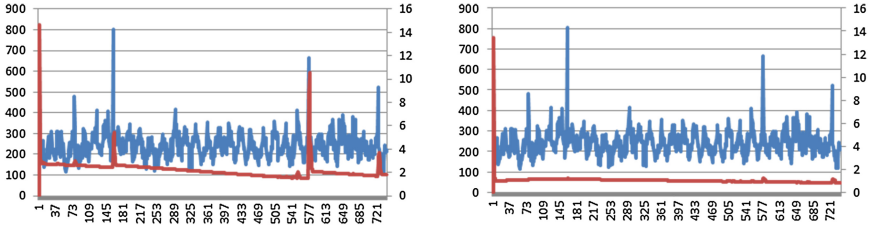

Fig. 3. Appropriate parameter set (left) and inappropriate one (right)

3.2 Coping with Many Runs: Multiple Query Optimization for CPD

As described above, to detect malware appropriately, multiple CPD procedures should be executed simultaneously. Let's think about a situation that we execute many CPD runs each of which has an identical parameter set. Then the amount of computation should be large. It naturally incurs performance degradation such as long latency. Such a long latency is not desired since delay of malware detection may increase damage to internal network by the intrusion.

A simple way is using advanced hardware such as many-core, FPGA or GPGPU that provide massively parallel computations. This is a promising approach. However, it requires additional power and money cost. This paper adopts a different approach. It is multiple query optimization that shares computations.

Moving time T is used for smoothing outlier score and computing change point score. Discounting parameter R and AR order K are used for the computation of SDAR algorithm. Discounting parameter R is only used for computation the mean μ in SDAR algorithm. Obviously, these parameters can be same. The inputs of the second stage learning are generated by the output of the first stage learning, and they are usually different. Therefore, it is difficult to share the computation of the second stage learning.

We denote discounting parameter, AR model order and moving time T of the first stage learning are denoted as α_R, α_K and α_T respectively. Similarity, we denote the inputs of second SDAR as β_R, β_K and β_T.

3.2.1 Sharing Multiple CPDs in Four Patterns

There are six parameters for CPD. Parameter α_R and α_K are used for CPD-1 (SDAR). Parameter α_T is used for CPD-2. Parameter β_R and β_K are used for CPD-3(SDAR). Parameter β_T is used for CPD-4. Depending on situation, we propose to share computations in four patterns as shown in Fig. 4. Assume we have two users, user 1 and user 2 who issue CPD queries. In naïve case, an input (x_t) is routed to inputs of two CPDs, CPD-1s. Even if parameters are the same, four steps (CPD-1, CPD-2, CPD-3, and CPD-4) are executed for each user. It should be noted that shared CPDs should form of tree. It is because all of child nodes must have the same input computation value from a parent. Our proposal reduces the execution cost by sharing computations as follows:

Pattern 1: Sharing CPD-1 if α_R and α_K are the same.
Pattern 2: Sharing CPD-1, 2 if α_R, α_K and α_T are the same.
Pattern 3: Sharing CPD-1, 2, 3 if α_R, α_K, α_T, β_R and β_K are the same.
Pattern 4: Sharing CPD-1, 2, 3, 4 if α_R, α_K, α_T, β_R, β_K and β_T are the same.

3.2.2 Sharing Multiple SDARs

As shown in Fig. 2, SDAR algorithm is constituted of five steps. They are SDAR-1, SDAR-2, SDAR-3, SDAR-4, and SDAR-5. SDAR-1 uses only discounting parameter (α_R for CPD-1 or β_R for CPD-3). The rest of steps use both discounting parameter and AR order (α_K for CPD-1 or β_R for CPD-3). Therefore, if discounting parameter is the same for two queries, then we share the computation to improve performance. We show additional sharing patterns in Fig. 4: pattern-1' and pattern-3'.

Pattern 1': Sharing SDAR-1 in CPD-1 if α_R are the same.
Pattern 3': Sharing CPD-1, 2 and SDAR-1 in CPD-3 if α_R, α_K, α_T, β_R are the same.

3.2.3 Choosing the Sharing Parameters

To improve the performance, the larger sharing is the better. Therefore our policy to choose sharing patterns is as follows.

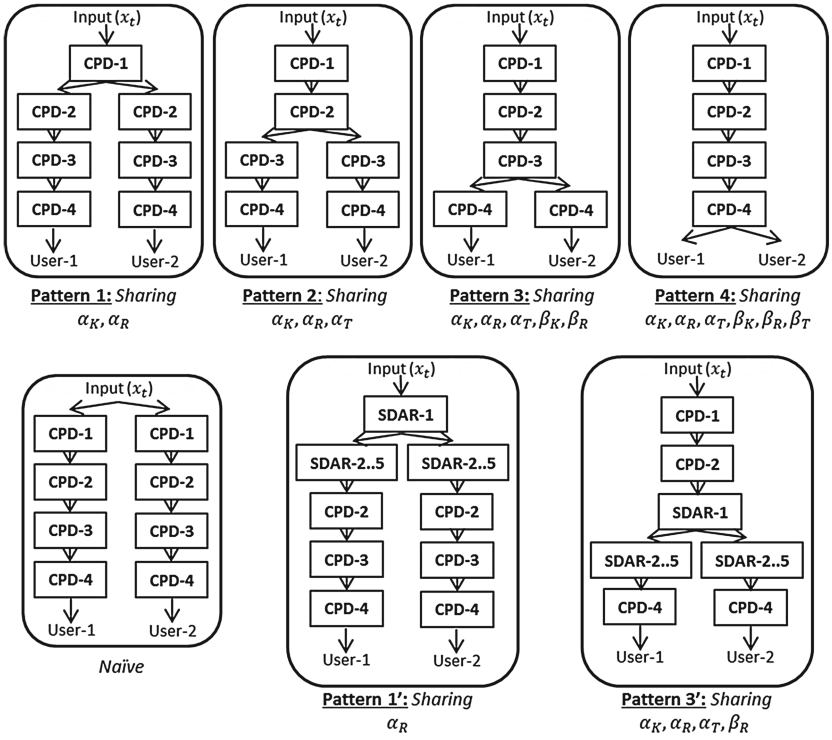

Fig. 4. Naïve computation and sharing computations for multiple runs

Algorithm. 1. Share of Parameters

```
1: BEGIN
2:   Create a cluster in level 0, "cluster-0-1";
3:   Add all of queries to cluster-0-1;
4:   FOREACH α_R, α_K, α_T, β_R, β_K and β_T
5:       Divide queries to multiple clusters so that all the
       queries in a cluster have the same parameter, and all
       the queries have the same parent;
6:   ENDFOR
7: END
```

Step 5 of algorithm 1 is important. When we have N queries, step 5 executes scan operation over K queries to find the same values in a cluster. Therefore its time complexity is O(N) for N queries (where N is the sum of queries in all of clusters) in a step. Since we have six parameters, total scan cost becomes 6 N.

4 Evaluation

Our experimental environment is as follows. OS: Ubuntu 10.04 LTS, CPU: Intel(R) Xeon(R) CPU E5640, 2.60 GHz, 4 cores, RAM 16 GB.

4.1 Performance Improvement by Number of Operators

We first investigated how much of micro operators can be reduced by using our proposal. Micro operator means a part of CPD or SDAR denoted as CPD-x or SDAR-x in the above. We tried three kinds of parameter sets. In the first parameter set, each parameter can be shared by all the queries. In the second parameter set, parameter values are provided randomly. In the third parameter set, parameter values are provided in the grid style, which requires additional explanation. Let's assume we have integer parameter x and y. Since both parameters can take infinite types, we should pick up some representative parameter sets. If we sample two points for each parameter, the number of parameter sets becomes 4 (= 2 × 2). Figure 5 shows x and y take 1 and 2 respectively, and parameter sets become {1,1}, {1,2}, {2,1}, and {2.2}. Grid style sampling is often used as a parameter set selection policy. The result of multiple runs are summarized using aggregation technique such as majority voting.

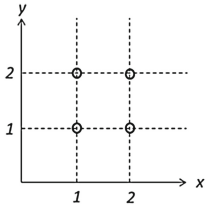

Fig. 5. Parameter sampling in grid style ({1,1}, {1,2}, {2,1}, and {2.2})

We show the result of experiments in Table 1. Parameter pattern used for the experiment was uniform, 3 random pattern sets, and 3 grid style sets. Applying multiple runs with different parameter sets and aggregating result is common in data mining [4]. However, to the best of our knowledge, appropriate tuning method for parameter sets is not yet matured. Obviously uniform dataset provides dramatic performance though such a case does not happen in the real. In random cases, performance gain is deeply related to types of random values. As types increase, performance gain reduces. In grid style cases, performance gain is also related to N (number of sampling values). As N increases, performance gain increases. Our proposal is especially effective for grid style policies.

Table 1. Reduction of operators by sharing techniques

Parameter pattern	# Queries	Naïve (# operators)	Sharing (# operators)	Performance gain (reduction ratio)
Uniform	64	384	6	98.4 %
Random (2 values)	64	384	101	73.7 %
Random (10 values)	64	384	315	18.0 %
Random (100 values)	64	384	366	4.7 %
Grid style (N = 2)	64	384	126	67.2 %
Grid style (N = 4)	4096	24576	5460	77.7 %
Grid style (N = 8)	262144	1572864	299592	80.1 %

It should be noted that that "performance gain" does not mean execution time. It means the number of reduced micro operators.

4.2 Performance Improvement in Execution Time

To measure the performance in execution time, we prepared a dataset. It is a data sequence generated by the following AR model.

$$x_t = 0.6x_{t-1} - 0.5x_{t-2} + \varepsilon_t \tag{1}$$

In the equation, ε_t is a Gaussian random variable with mean 0 and variance 1. This dataset consists of 10000 records. Change points occur at times $1000 \times k + 1 (k = 1, \ldots, 9)$.

We consider a situation that we apply 100 CPDs simultaneously to the dataset. We measure execution time for both our proposal and naïve policy. Here we explain the meaning of symbols in Table 2. α_K and β_K expresses AR order. α_T and β_T express averaging time. α_R and β_R express discounting parameters.

4.3 Result

Table 2 shows the result of experiments. "Shared CPD-1" denotes a case that shares multiple CPDs when $\alpha_R, \alpha_K, \alpha_T$ are the same can be shared. "Shared SDAR-1" denotes a case when only α_R can be shared. We chose 10 patterns of parameter sets as shown in the Table 2.

Table 2. Execution time of naive, shared CPD-1 and shared SDAR-1.

ID	Parameters						Execution time (s)			Performance gain (times)	
	α_R	α_K	α_T	β_R	β_K	β_T	Naive	Shared CPD-1	Shared SDAR-1	Shared CPD-1	Shared SDAR-1
1	.02	2	5	.02	3	5	2.92	1.77	2.84	1.65	1.03
2	.02	4	5	.02	3	5	3.65	1.77	3.58	2.06	1.01
3	.02	2	5	.02	4	5	3.29	2.17	3.22	1.52	1.02
4	.005	2	5	.02	4	5	2.91	1.76	2.82	1.65	1.03
5	.02	2	5	.005	3	5	2.89	1.77	2.82	1.64	1.03
6	.02	2	7	.02	7	5	3.00	1.87	2.88	1.60	1.03
7	.02	1	5	.02	1	5	1.96	1.11	1.88	1.78	**1.04**
8	.02	**10**	10	.02	10	10	11.2	5.84	11.1	**1.92**	1.01
9	.02	1	10	.02	10	10	6.66	5.80	6.61	1.15	1.01
10	.02	**10**	10	.02	1	10	6.68	1.34	6.57	**5.00**	1.02

We found that AR order is an important factor for performance. Experiment 10 shows that "shared CPD-1" is 5 times faster than "naïve", which is the best performance improvement. In this case, AR-order α_K is set to 10, which is the highest in parameter sets. An interesting case is experiment 8 which achieves at most 1.92 times

improvement though AR-order α_K is 10. The reason may be because of latter part. Since second AR-order β_K is also set to 10 in experiment 8. Our experimental condition does not execute sharing in the latter part, and therefore larger β_K incurs performance degradation indicated as the worst execution times which are 11.2 s for Naïve, 5.84 s for shared-CPD1, and 11.1 sec for shared-SDAR-1 respectively.

Another observation is small improvement gained by sharing SDAR-1. Performance improvement by SDAR-1 is at most 4 %, which is negligible when comparing with CPD-1.

5 Conclusions and Future Work

This paper proposed to incorporate a multiple query optimization scheme for change point detection. We found 6 sharing patterns for CPD. The result of experiments showed that our scheme reduces 80.1 % of operators when parameter setting policy is grid style (N = 8), and achieves 5 times performance improvement compared with naïve approach. This paper concludes that our scheme for CPD is effective for at least synthetic datasets.

There are some future works. First future work is investigating the degree of effectiveness using real datasets including packet streams including malware. Second future work is reducing the cost to decide whether sharing or not. Our current algorithm shown in Algorithm 1 takes O(N), which is too large then N is enormous.

Acknowledgement. This work is supported by KAKENHI(#24500106).

References

1. I, D., Yoshioka, K., Eto, M., Yamagata, M., Nishino, E., Takeuchi, J., Ohkouchi, K., Nakao, K.: An incident analysis system NICTER and its analysis engines based on data mining techniques. In: Köppen, M., Kasabov, N., Coghill, G. (eds.) ICONIP 2008, Part I. LNCS, vol. 5506, pp. 579–586. Springer, Heidelberg (2009)
2. Takeuchi, J., Yamanishi, K.: A unifying framework for detecting outliers and change points from time series. IEEE Trans. Know. Data Eng. **18**(4), 482–492 (2006)
3. Madden, S., Shah, M., Hellerstein, J.M., Raman, V.: Continuously adaptive continuous queries over streams. In: Proceedings of ACM SIGMOD, pp. 49–60 (2002)
4. Li, H., Sun, J.: Majority voting combination of multiple case-based reasoning for financial distress prediction. Expert Syst. Appl. **36**(3), 4363–4373 (2009)
5. Srivastava, D., Golab, L., Greer, R., Johnson, T., Seidel, J., Shkapenyuk, V., Spatscheck, O., Yates, J.: Enabling real time data analysis. Keynote Proc. VLDB Endowment (PVLDB) **3**(1), 1–2 (2010)