Two-Steps Independent Approach for Rule-based Complex Event Processing

Sunyanan Choochotkaew[1,a)] Hirozumi Yamaguchi[1,b)] Teruo Higashino[1,c)]

Abstract: Rule-based solutions are broadly applied for matching event-pattern in Complex Event Processing (CEP). Most of the existing works store and process all rules converted from event definition as a single set and activate irrelevant copies of partially detected events every time new data comes. Against this issue, we analyze the processing time of using one single set of rules and using two-steps matching where one complex rule is separated and processed by two modules independently. The experimental results show that the general single rule set processing can cause significant delays when some temporary events stay in the working memory in long time comparing to two-steps approach.

1. Introduction

Now, various kinds of software applications can provide much higher benefit from analyzing a large amount of information which is usually fed by multiple sources. Not only to respond to human demands as quickly as required, it is beneficial to perform trend analysis as soon as the data are available. According to [3], continuously flowing data can be considered by Stream Processing model or Complex Event Processing (CEP) model. Nevertheless, the definition or boundary between CEP and Stream Processing model is still unclear and not deeply discussed. Recently, the word CEP is likely to cover all studies in the information processing area.

In CEP, one of the most important part is extracting the interesting information from such a continuous and high-volume, sometimes various, flows. Many researchers exploit the matching power of rule-based engines for solving this part [5],[1],[6]. They match the event-defined language which is easy-to-understand to the complicated engine rules called Rule-based CEP. Straightforwardly applying rule-based engines for complex event detection in the earlier works needs some concerns. Generally, a complex event is composed of multiple simple events which can be detected at the first step. These simple events may overlap between multiple complex events and may repeatedly detected. This situation incurs unnecessary computational costs. The most trivial solution is detecting all simple events and making a copy of the detected one for each referencing complex event. Doing that, the engine is required to keep the temporary copies in the working memory waiting for the further detection.

Most of rule-based publications defines one rule set and keep all copies in the same working memory. In other word, hardly detected rules and frequently detected rules share the same working memory. Consequently, a temporary copy of partial detected event will be called to test the composite rules every time when facts or rules changed even if it has no relevance. We concern that this type of overhead is going to be crucial for future IoT processing that may be done on more tiny computers. In this paper, we analyze the processing time affected by the staying time of temporary copies for complex detection in the working memory between using one single rule set (i.e. dependent working memory) and using separate rule sets (i.e. independent working memory).

2. Two-Steps Approach

Performance of rule-based engine basically depends on the number, complexity, and arrangement of rules and number of facts. In the real-time processing, piling up the copies of partial detected events which are considered as facts for complex event detection in the same working memory can significantly slow down the matching process. Because a temporary fact has to be called whenever a new fact comes, which is usually continuous. The additional cost corresponds to the staying time which can be determined from the sparseness of all matching events in the flow. The staying time of temporary copy $subCE_i$ detected at time t may be primarily approximated from the following equation where x_i is the number of referenced simple sub-events in complex event, (CE), $count(subCE_k)$ is the number of each of the other relevant sub-events remaining in the flows at time t, and $n(t)$ is the number of events in the flow at time t:

1 Graduation School of Information Science and Technology Osaka University, Osaka, Japan
a) sunya-ch@ist.osaka-u.ac.jp
b) h-yamagu@ist.osaka-u.ac.jp
c) higashino@ist.osaka-u.ac.jp

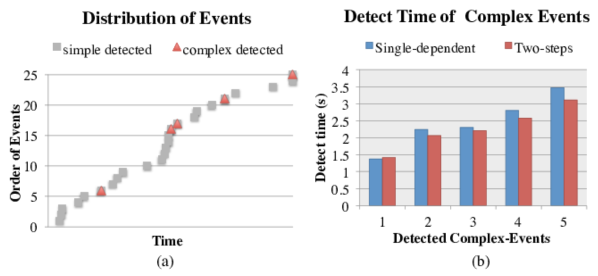

Fig. 1 (a) Distribution of simple events in the experiment flow (b) Bar graph comparing detected time of complex event between single-dependent and two-steps approach

$$StayingTime(subCE_i(t)) \approx \sum_{k=1,k\neq i}^{x_i} \frac{count(subCE_k(t))}{n(t)}$$

Notice that, a single dependent solution always activates temporary copies in all staying time to process continuously flowing information.

To reduce the activate time, we suggest a two-steps independent approach which keep the temporary fact in the separate working memory and call it only when another temporary fact, which is supposed to be less frequent, is asserted. So, the number of activating time will be the number of detected sub-event between t and $t + StayingTime$ which is less than all sampling during StayingTime. In the most simple way, we can apply content filtering (e.g. Simple Rule: **temperature** $> 45 \rightarrow ExtremeTemp$) as first step before continuing the rest complex processing (e.g. Complex Rule: **ExtremeTemp** *within* 5 *min from* **Smoke** $\rightarrow$ *Fire* [2]) in the second step.

3. Preliminary Results

To measure the processing time of the real engine, we deploy our two-steps independent concept on the JESS rule-based engine implemented in JAVA[4] with three sets of 10000 dumb information from 20 flows. We defined a complex event which is composed of five simple events. There are five complex events supposed to be detected from the experiment flows.

See Fig. 1, the left figure (a) shows distribution of the simple events in the flow and points that complex events must be detected while the right figure (b) shows the corresponding detected time comparing single-dependent and two-steps approach. At the first detection, simple events are highly-frequently detected. There are just small amount of temporary simple facts and detected time of both approaches are slightly different. At the third detection, complex events are consecutively detected. Difference of time slightly changes. For the second, forth, and fifth detection, simple events are infrequently detected. A number of temporary facts are accumulated. In this case, we can observe that Two-steps approaches can detect a complex event faster.

4. Conclusion

In this paper, two-steps dependent procedure is considered and analyzed for efficient implementation of rule engines in Complex Event Processing (CEP). The preliminary results implemented with the well-known rule-based engine, JESS, show that two-steps approach gains an advantage when the number of interesting complex events is very small and some partial parts stuck in the engine memory.

Acknowledgements

This work was supported in part by JSPS KAKENHI JP15H02690 and JP16KT0106.

References

[1] Cugola, G. and Margara, A.: RACED: An Adaptive Middleware for Complex Event Detection, *Proceedings of the 8th International Workshop on Adaptive and Reflective MIddleware*, ARM '09, NY, USA, ACM, pp. 5:1–5:6 (2009).
[2] Cugola, G. and Margara, A.: TESLA: A Formally Defined Event Specification Language, *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, DEBS '10, NY, USA, ACM, pp. 50–61 (2010).
[3] Cugola, G. and Margara, A.: Processing Flows of Information: From Data Stream to Complex Event Processing, *ACM Comput. Surv.*, Vol. 44, No. 3, pp. 15:1–15:62 (2012).
[4] Friedman-Hill, E.: Jess, The Rule Engine for the Java Platform, `http://herzberg.ca.sandia.gov` (2007).
[5] Li, G. and Jacobsen, H.-A.: Composite Subscriptions in Content-based Publish/Subscribe Systems, *Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware*, Middleware '05, NY, USA, Springer-Verlag New York, Inc., pp. 249–269 (2005).
[6] Li, G., Muthusamy, V. and Jacobsen, H.-A.: *Middleware 2008: ACM/IFIP/USENIX 9th International Middleware Conference Leuven, Belgium, December 1-5, 2008 Proceedings*, chapter Adaptive Content-Based Routing in General Overlay Topologies, pp. 1–21, Springer Berlin Heidelberg (2008).