Learning Process Models in IoT Edge

Long Cheng[†], Cong Liu[*], Qingzhi Liu[*], Yucong Duan[‡], and John Murphy[†]
[†]School of Computer Science, University College Dublin, Ireland
[*]Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands
[‡]College of Information Science and Technology, Hainan University, China
long.cheng@ucd.ie, c.liu.3@tue.nl, q.liu.1@tue.nl, duanyucong@hotmail.com, j.murphy@ucd.ie

***Abstract*—Process models as knowledge graph representation have been widely used in various domains to create products and deliver services. Although different process model discovery approaches have been proposed in recent years, few of them are designed for distributed computing environments. Specifically, none of them has been studied in the emerging edge computing application scenarios. In this paper, based on the requirements of some real-time process services, we propose a system design for learning process models in IoT edge. We present the details of our solution and our preliminary results on a simulated IoT network show that our method can discover real-time process models in less than a second.**

***Index Terms*—process mining; model discovery; edge computing; IoT; service computing**

I. Introduction

Process mining is an active research discipline aiming at extracting non-trivial knowledge and interesting insights about processes from event logs. As one of its core tasks, process discovery takes an event log as input and produces a process model without using any prior information [1]. Such a discovered model can be seen as a knowledge graph, which abstracts the behavior of all the events in the event log and also represents all their possible execution semantics. Process models have been widely used in various domains, such as to create products and deliver services.

We focus on learning process models from event data generated from Internet of Things (IoT). Specifically, we are interested in discovering and delivering process models as a service in IoT enviroments in real-time where possible. In this case, data consumers will be able to conduct online deviation analysis and abnormal detection. The commonly used discovery algorithms such as the Inductive Miner [2] can not be applied in such scenarios directly. The reason is that these approaches focus on processing event logs in a centralized way. Within such a scheme, aggregating event data from large IoT networks to a standalone machine is costly. Moreover, model discovery will be also time consuming when the number of events is large. We can deploy a distributed process model discovery algorithm like the one using MapReduce [3] in a cloud to speed up the data aggregation and computing process. However, such kind of cloud-based computing has been shown to be not efficient enough for many data applications, e.g., the ones require very short response time [4].

To meet the real-time service requirements on process model discovery and delivery in IoT, we propose a design by leveraging the emerging edge computing paradigm. In fact, edge computing has been shown to be a better solution for processing time-sensitive events on supporting instantaneous response and subsequent decision making, compared to cloud computing. The main reason is that edge computing brings memory and computing power closer to the location where it is needed. Instead of sending data to a centralized cloud repository or a central data center, data processing happens at a local gateway device (i.e., edge server) in edge computing. In this case, the network pressure can be greatly reduced and consequently the service response time can be improved.

In general, we focus on efficient process model discovery in IoT edge for real-time services. This is the first time on how to apply process mining technique in edge computing. Our main contributions can be summarize as follows.

- We introduce a system design for learning process models in edge computing, aiming at discovering and delivering process models as a service to data consumers in real-time.
- Based on the cluster-based structure of IoT networks, we propose an efficient discovery approach supporting both intra-cluster and inter-cluster process model discovery on edge servers.
- We conduct an experimental evaluation of our method in a simulated IoT network and the results demonstrate our approach is efficient on process model discovery.

The paper is organized as follows. Section II defines some terminologies and basic notations. Related work are introduced in Section III. The proposed system architecture and discovery approaches are presented in Section IV. We present our preliminary experimental results in Section V and conclude the paper in Section VI.

II. Preliminary

To obtain process models, we rely on event logs. An *event log* can be considered as a multi-set of traces [5]. Each *trace* is a finite sequence of *event*s where an event refers to an *activity*. An event log can be considered as a multiset of traces because there can be multiple cases having the same trace.

Definition 1. ***(Trace, Event Log)*** *Let $\mathcal{A}$ be a set of activity labels. A trace is a sequence of activities, i.e., $\sigma \in \mathcal{A}^*$. An event log is a multiset of traces, i.e., $L \in \mathbf{B}(\mathcal{A}^*)$.*

Given an example log $L_1 = [\langle a, b, c, d, e\rangle^{20}, \langle a, c, b, d, e\rangle^{10}]$. It contains 30 traces with 20 traces following $\langle a, b, c, d, e\rangle$ and 10 traces $\langle a, c, b, d, e\rangle$.

As Petri nets [5] are capable of combining a graphical representation and a formal foundation, they have been widely used to model, analyze and verify business processes. In this work, we use Petri nets to represent process models. Some of the essential terminology and notations regarding Petri nets are presented as follows.

Definition 2. ***(Petri net)*** *A* Petri net *is a 3-tuple* $PN = (P, T, F)$*, satisfying (1)* $P \cap T = \emptyset$*,* $P \cup T \neq \emptyset$ *where* P *is the place set and* T *is the transition set; and (2)* $F \subseteq (P \times T) \cup (T \times P)$ *is the flow relation.*

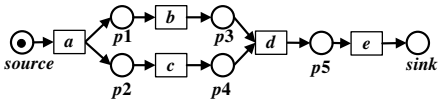

Fig. 1. An Example Petri Net

A marking M is a multiset of places, i.e., $M \in \mathbf{B}(P)$. The state of Petri net, called *marking*, is a multiset of places indicating how many tokens each place contains. Fig. 1 shows an example Petri net where $[source]$ is its initial marking. A transition $t \in T$ is *enabled* in marking M, denoted as $M[t>$ if each of its input places $^\bullet t$ contains at least one token. Consider the example Petri net in Fig. 1 with $M = [p_3, p_4]$, $M[d>$ because both input places (p_3 and p_4) are marked. An enabled transition t may *fire*, i.e., one token is removed from each of its preset $^\bullet t$ and one token is added for each of its postset $t^\bullet$. Additionally, we consider execution sequences of a *PN* from an initial marking and a final marking.

Definition 3. ***(Process Discovery Algorithm)*** *Let* $\mathscr{U}_{\mathscr{P}}$ *be the universe of all process models. A process discovery algorithm is a function* γ *that maps an event log* $L \subseteq \mathcal{A}^*$ *to a process model* $pm \in \mathscr{U}_{\mathscr{P}}$*, i.e.,* $\gamma(L) = pm$*.*

Generally speaking, a process discovery algorithm γ is capable of converting an event log to any form of process models, such as *Petri net*, *Business Process Modeling Notation*, *Event-driven Process Chain*, etc. Whatever representation is used, each trace in the input event log corresponds to a possible execution sequence in the discovered process model. Consider for example, by taking as input L_1, one can discover a Petri net that looks like the one in Fig. 1. All traces in L_1 can be replayed by the Petri net in Fig. 1, i.e., each trace corresponds to an execution sequence in the net.

III. Related Work

Process discovery is one of the most important tasks in process mining. The problem on how to discover a good process model and how to learn the model in an efficient way are still challenging current discovery approaches. Many disovery algorithms such as the α-miner [6] and Inductive Miner [2] can learn process models from event logs in which each process instance is recorded as a case with ordered events. However, all the methods are on the basis of a standalone implementation and thus they can not be applied in distributed computing environments directly.

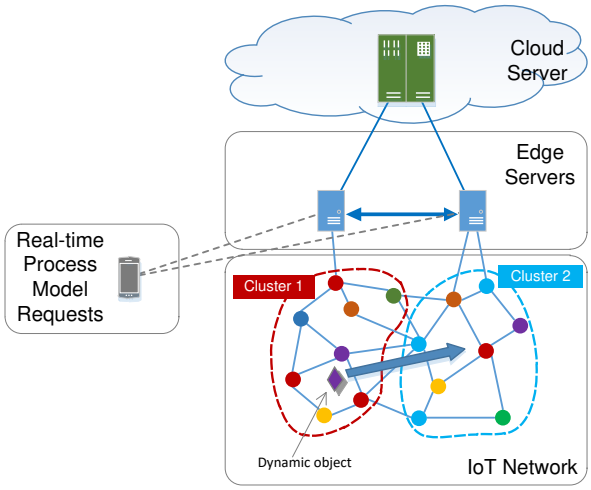

Fig. 2. The system architecture of process model discovery in edge computing. The data aggregation is based on the partitioned clusters in IoT networks. The discovered models in edge servers are delivered to terminals in real-time based on service requirements.

With the growth of IoT networks, the quantity of generated event data has posed great challenges for current process mining techniques [7]. For the purpose of efficient process model disovery over large event logs, the state-of-the-art approach [3] uses MapReduce leveraging distributed computing platforms. Its core idea is to distribute logs over computing nodes, so that the discovery tasks can be efficiently computed in parallel. Experimental results have shown that the method can achieve obvious speedups compared to a standalone implementation. From a service computing aspect, the alogrithm can be deployed in cloud to improve service efficiency. However, transmission of the data to a cloud will bring in obvious overhead for real-time services. Moreover, a process model in a specified area is more interesting sometimes, and computing such a model over the data collected from the whole IoT netowrk in a cloud scenario will be costly in terms of both computing resouces and time. Recently, a hybrid process model discovery method has been proposed for handling large event logs [8]. Similar to [3], the method is designed for cloud computing, and thus it can not be applied to the case presented in this work.

IV. Process Model Discovery in Edge Computing

In this section, we describe our system design and approach for learning process models in edge computing.

A. System Architecture

The system architecture includes an IoT network and edge computing servers as shown in Fig. 2. The IoT network consists of heterogeneous nodes (e.g., sensors) with different functions (in different colors). Event data is produced by each node either periodically or triggered by dynamic objects around the node, and the generated events are transferred to

edge servers by multi-hop communication. The edge servers can make either parallel or distributed computing for the aggregated event data from the IoT network. These servers can re-allocate the data to multiple servers through high bandwidth network, such as WiFi, 5G or wired network, which can easily achieve Gigabit communication speed. In a general case, we utilize a cluster-based data collection solution in the IoT network [9]. The IoT network has been partitioned into clusters, e.g., each cluster means the sub-network deployed in a large organization or a city. For the simplicity of the presentation, we assume there is only one edge server in each cluster, which is responsible for collecting the data from the IoT nodes in the cluster.

B. Process Model Discovery

The data operations for model discovery in our system mainly contain two layers as follows.

1) Trace Construction: The event data generated by each IoT node is transferred to the assigned edge server in real-time, and the server constructs traces based on the events in a dynamic way. To focus on process model discovery, we assume that each generated event is associated with a case id. For a specified condition where there is no case id, we can use relevant techniques such as event correlation [7] to discover a case id. To provide a real-time model for consumers, the constructed traces will be updated batch by batch and only the ones in a specified time window will be used for model discovery. The events which are not in the window will be considered as historical data and transferred to cloud if required, for storage or global analysis.

2) Model Discovery: For a service requirement on a single real-time process model, our discovery will perform on the events from either a single edge server or multiple ones. We call the former case as *intra-cluster discovery*, because its discovered model can only describe the event behavior in a specified IoT cluster. Similarly, we name the latter case as *inter-cluster discovery*, for the purpose of discovering a model to represent the behavior of dynamic objects, which could move cross different clusters and trigger the relevant IoT nodes, as demonstrated in Fig. 2. As these two discovery implementations are the most basic tasks for model discovery in edge computing, we will focus on these two cases in this work. For many other complex service requests, for example, concurrent service requests on different models (either intra or inter-cluster) from multiple users in different locations, we can adopt advanced task scheduling strategies over the edge servers to achieve the best possible performance on discovery and to meet the service level agreements.

a) Intra-cluster discovery: For the intra-cluster discovery, all the required events will be on a single edge server. Therefore, we can use existing process model discovery algorithms on the server to discover a process model. In this paper, we use the *inductive miner* [2], as it is currently one of the leading process discovery techniques that can guarantee correctness of discovered process models.

The inductive miner discovers process models using a divide and conquer approach. Given an input event log, it searches for possible log splits repeatedly such that the original log is split into smaller sub-logs. The split is built on top of the so-called directly-follows graph that captures direct succession information of the activities in an event log. The cut found from the directly-follows graph is used to split the corresponding log into smaller ones. This step iterates until a log contains only a single activity or no cut can be found. Four types of cuts that correspond to four basic process patterns, sequence, concurrency, choice and loop, are defined. More explanations of inductive miner are referred to [2].

b) Inter-cluster discovery: Because of the dynamics of objects, triggered events with a same case id could be distributed over different clusters in an IoT network. To discover an inter-cluster model, we need to collect all the relevant cross-cluster events on a single edge server and then use the inductive miner on model discovery. To identify these events for a given service request, we use an additional table on each server to record its local case ids in real-time. Once receiving an inter-cluster discovery request, the edge server will communicate with the relevant servers to (1) check whether its recorded case ids appear on the servers, and (2) retrieve the relevant constructed sub-traces if the ids appear.

For example, we have a request on the edge server X (or cluster X), which aims to get an inter-cluster model with its neighbored cluster Y. Assume that at such a time point there is a trace $\langle a, b\rangle$ with case id 3 on X and a trace $\langle c, d, e\rangle$ with case id 3 on Y, then the edge server X will retrieve its required data $\langle c, d, e\rangle$ from Y to construct a local trace $\langle a, b, c, d, e\rangle$ with case id 3 for model discovery.

C. Comparison to Current Solutions

Taking a high level comparison with the standalone algorithms and the state-of-the-art cloud-based implementation [3], there are two main advantages to our design in providing real-time process model services: (1) From a system performance aspect, we do not need to aggregate all the event data to either a single machine or a centralized cloud repository for process model discovery. In our system, each edge server collects data independently and they only need to share part of their data for cross-cluster objects when performing inter-cluster discovery. This makes computing (i.e., model discovery) closer to the data sources and thus can greatly reduce the network pressure and consequently improve real-time performance; (2) From a service angle, our distributed architecture has also put services closer to data consumers, which can also improve real-time efficiency. Moreover, for concurrent requests on local models (either based on the data from a cluster or multiple clusters), the model discovery in our system will perform in a more *distributed* way rather than a *parallel* way. Namely, each edge server is responsible to discover an independent model in parallel rather than all the edge severs compute a single model in parallel at a time. This can remove the overhead of starting parallel programs and communication synchronization in a cloud-based system.

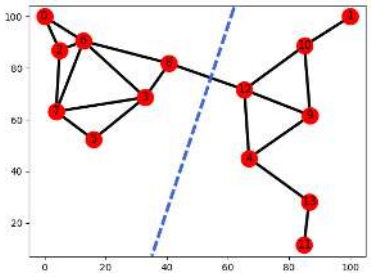

Fig. 3. The simulated IoT network with edge servers. The network is partitioned into two clusters as demonstrated by the dashed line.

V. Preliminary Experimental Results

We evaluate the performance of the proposed discovery approaches in a simulated IoT network. As demonstrated in Fig. 3, the network is deployed in a square area of 100m$\times$100m. As a preliminary experiment, we test our solution using 11 sensor nodes. The nodes are randomly scattered in the area. The network is partitioned into two clusters. Each IoT node sends data to the edge server of its residing cluster. We suppose the IoT nodes transmit data to the edge servers by multi-hop communication, and use the shortest path as the route from each IoT node to the edge server. We set 100 dynamic objects (e.g., robots) in the IoT area, and each object moves following some specified patterns. When an object reaches an ending point, it will move back to its start point to start a new moving pattern. We set the node 0 and 1 in Fig. 3 as the two edge servers, and the other nodes with numbers $\{2, 3, ..., 13\}$ are IoT nodes, which are responsible to the activities $\{A, B, ..., L\}$ respectively. The speed of each object is 20m/s and the communication speed between IoT nodes is 1Kb/s. To simplify the simulation, we ignore the interference in wireless communication.

We assume that we have a service request on the edge server 0 for an intra-cluster and an inter-cluster process model every 30 seconds. We report the experimental results at the time point that the system has run for 90 seconds. Namely, our model discovery will be based on the received (and retrieved) events on the edge server 0 between the time points 60 and 90 second. Fig. 4 shows the discovered intra-cluster process model, and Fig. 5 demonstrates the discovered inter-cluster process model. Both the discovery executions are done in 0.3 second including remote data retrieving. In this light, we believe that our edge based solution has the potential capability to deliver high-quality process models as a service in real-time.

VI. Conclusion and Future Work

This paper presents a solution for learning process models in IoT networks with edge servers, which aims to deliver process model services in real-time. We give the detailed system architecture design and describe the intra-cluster and inter-cluster process model discovery approaches in our solution. Our preliminary experiments based on a simulated IoT environment show that our approach can discover both intra and inter-cluster process models in less than a second.

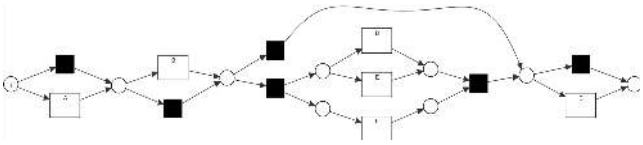

Fig. 4. The discovered intra-cluster process model on edge server 0 at the time point 90 sec.

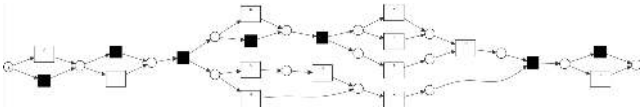

Fig. 5. The discovered inter-cluster process model on edge server 0 at the time point 90 sec.

Our future work mainly lies in extending our method to handle more complex service requests, e.g., concurrent service requests, and more IoT-related business processes, e.g., cross-organization transportation business processes [10]. Moreover, we plan to use more advanced strategies to further improve the real-time efficiency of our method. For the inter-cluster model discovery, instead of aggregating events, we will try to aggregate the directly-follows relationships of all the relevant cross-cluster events. In this case, the computation on directly-follows relationships will be able to be done in parallel by the involved edge servers. Finally, we will evaluate the real-time performance of our solution in large IoT networks.

Acknowledgments

This work is supported by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 799066.

References

[1] C. Liu, Y. Pei, Q. Zeng, and H. Duan, "LogRank: An approach to sample business process event log for efficient discovery," in *Proc. 11th Int. Conf. Knowl. Science, Eng. Mgmt.*, 2018, pp. 415–425.

[2] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from event logs-a constructive approach," in *Int. Conf. Applications and Theory of Petri Nets and Concurrency*, 2013, pp. 311–329.

[3] J. Evermann, "Scalable process discovery using Map-Reduce," *IEEE Trans. Serv. Comput.*, vol. 9, no. 3, pp. 469–481, 2016.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[5] W. van der Aalst, *Process Mining: Data Science in Action*, 2016.

[6] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, 2004.

[7] L. Cheng, B. van Dongen, and W. van der Aalst., "Efficient event correlation over distributed systems," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud and Grid Comput.*, 2017, pp. 1–10.

[8] L. Cheng, B. van Dongen, and W. van der Aalst, "Scalable discovery of hybrid process models in a cloud computing environment," *IEEE Trans. Serv. Comput.*, pp. 1–1, 2019.

[9] Q. Liu, T. Ozcelebi, L. Cheng, F. Kuipers, and J. Lukkien, "Cluflow: Cluster-based flow management in software-defined wireless sensor networks," in *IEEE Wireless Comm. and Networking Conf.*, 2019.

[10] C. Liu, H. Duan, Z. Qingtian, M. Zhou, F. Lu, and J. Cheng, "Towards comprehensive support for privacy preservation cross-organization business process mining," *IEEE Trans. Serv. Comput.*, no. 1, pp. 1–1, 2016.