Exploiting Process Knowledge for Event Processing in Distributed Business Applications

Holger Ziekow*

International Computer Science Institute
1947 Center Street, Berkeley, CA 94704, USA
ziekow@icsi.berkeley.edu

Abstract. In this paper, we address event processing for applications in distributed business processes. For this application context we present an approach for improving in-network processing of events. We highlight the role of a priori process knowledge for query optimization and describe distributed event processing based on decision points in the process.

1 Introduction

Event-based interaction plays a central role in a number of emerging distributed business processes. A prominent example is monitoring logistic events in supply chains. In such applications we can observe two trends: (1) emerging technologies like RFID and smart sensors increase the level of detail for event capturing; and (2) globalization increases the dynamics and complexity of value chains. The value chain processes spread out across countries and organizations and partner relations change frequently.

Business applications face the challenge to efficiently handle events in this environment. The foremost technological challenges are (a) the inherent distribution of event sources, (b) the large number of events, and (c) the dynamics of the underlying business processes. Distributed event-based systems (DEBS) provide the technological basis for addressing these challenges. They allow decentralizing tasks of event dissemination and processing. In this paper we tailor technologies of DEBS to applications where events are generated by business processes. By exploiting process knowledge we are able to realize optimizations that go beyond optimizations in generic event-based system.

The remainder of the paper is structured as follows. In Section 2 we review related work. Section 3 provides background information about the application context. It introduces an exemplary system setup from the RFID domain that we will use for illustrating our approach. Section 4 presents a mechanism for using process knowledge in distributed event processing and illustrates the benefits along a basic example. Section 5 concludes the paper and outlines future work.

* The author's second affiliation is the Institute of Information Systems at the Humboldt-Universität zu Berlin.

R. Meersman, P. Herrero, and T. Dillon (Eds.): OTM 2009 Workshops, LNCS 5872, pp. 277–281, 2009.

2 Related Work

Over the past few years, technologies for event processing gained increasing attention in the research community [2,3]. Most relevant to our approach is existing work on in-network processing. Solutions for in-network processing are concerned with deploying operators into a network of processing resources. Much work has been done in the field of sensor networks. Examples from this domain are TinyDB [4] and HiFi [5]. These systems aim to apply operators in the sensing network for reducing energy consuming communication. Srivastava et al. optimize operator placement for hierarchical networks of heterogeneous devices [8]. Pietzuch et al. use network coordinates for efficient operator placement in networks of large scale [7].

Due to space limitations we refrain from an more exhaustive discussion of related work. However, all approaches to our knowledge optimize operator deployment solely based on the query and some information on the processing network. Our work additionally uses process knowledge and thereby introduces new measures for optimization.

3 Events in Collaborative Business Applications

A business event is a happening of interest in the business world [6]. In collaborative value chains, the processes are distributed across several locations that belong to different organizations. Collaborating companies exchange business events to streamline their processes. A prominent example is an RFID-enabled supply chain, where supply chain partners exchange events to monitor the shipment process. We will use this example for illustration throughout the paper.

The push based exchange of RFID related events is supported by the EPCIS standard [1]. This standard allows for ad-hoc queries as well as standing queries for RFID events. Companies can query the EPCIS of their partners to capture RFID events along the supply chain. The underlying event model is based on the Electronic Product Code (EPC) that provides a globally unique numbering scheme for physical objects. The event model allows associating the observation of an object with a timestamp, a location, and the corresponding business step.

The EPCIS interface enables continuous queries with filters on the event attributes. However, correlation of events from different sources is not supported. Monitoring business distributed processes therefore requires event correlation at the query sink (i.e., the monitoring application). This architecture is inefficient compared to solutions that support processing in the network [7]. We therefore propose to use a processing network that links event sources with applications (see Figure 1). The network provides distributed resources that can run parts of the monitoring query. It has been shown that such an architecture improves efficiency [7]. Our work targets query processing in such an infrastructure. We subsequently show how we can exploit particularities of the application domain to optimize distributed event processing.

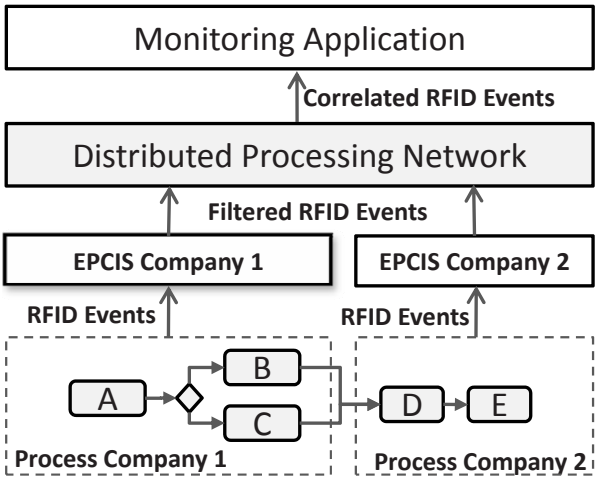

Fig. 1. System architecture

4 Optimization with Process Knowledge

In this paper we describe how to exploit timing information and decision points in process models for query optimization. Using this a priori process knowledge allows optimizing event processing for each instantiation of a process. Our goal is to optimize the query deployment. That is, we optimize the mapping of query operators to processing resources and the data flow between them.

Optimizing a query deployment for a certain process instance necessarily yields better results than a generic deployment which considers all options for the process. However, when the process starts, it is not known what the instance will look like; i.e. how decisions will be made at decision points. We must therefore evaluate decision points as the process runs, in order to consecutively align the deployment to the current process instance. We can thereby come close to an optimal query deployment for each process instance.

The basic idea is as follows: Passing a decision point during process execution narrows down the options for how the process continuous. To exploit this, we deploy a query such that evaluation of the first decision point in the process is possible. We then direct event messages according to the remaining options for the process. We thereby adapt distributed processing to the current process instance.

For illustration consider the process in Figure 2. It shows a simple process that is distributed across five organizations. Instantiations of this process can create event sequences of the form (A,B1,D1), (A,B1,D2), or (A,B2,D2). Further consider the simple sequence query A→B→D2 that matches process instances of the form (A,B1,D2) and (A,B2,D2). Without process knowledge, one must register for all event sources and detect the instances that end with D2 (see operator 1 in Figure 3, left side). However, we can exploit process knowledge as

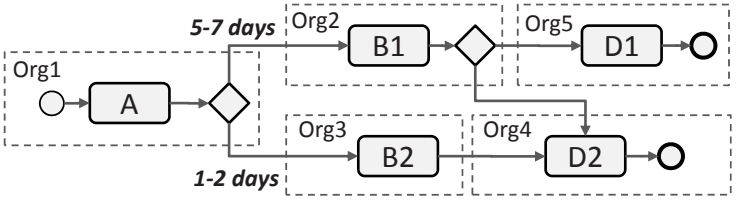

Fig. 2. Sample process

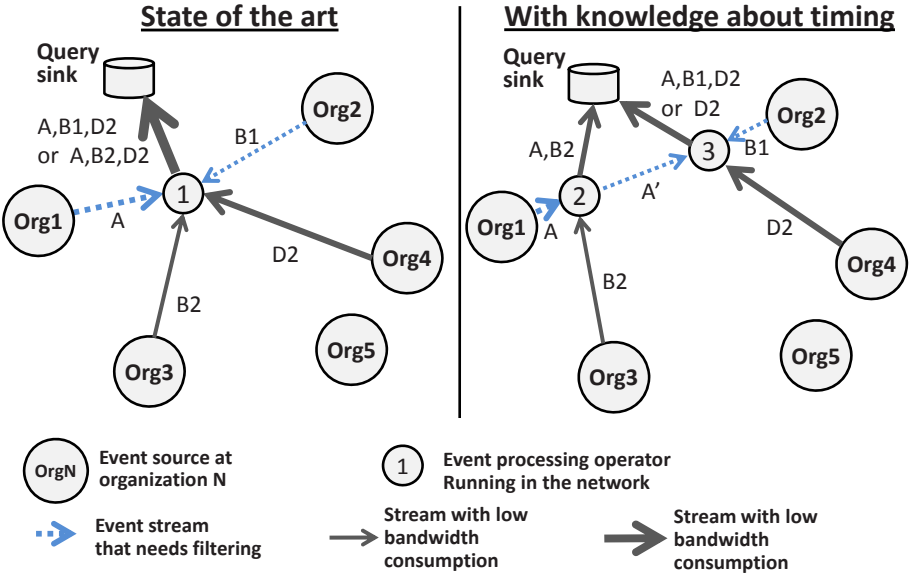

Fig. 3. Query deployment with/without process knowledge

follows: We first detect the decision at the first decision point in the process. We therefore simply register a query at A and B2 (see operator 2 in Figure 3, right side). Given the process knowledge, we can infer that the current instance will generate the event sequence (A,B2,D2) if B2 follows A within 2 days. At this point we already know that the query will match and send the events A,B2 right to the sink. If we get a timeout for B2, we know that process may generate the sequences (A,B1,D1) or (A,B1,D2) and further processing is required. We then forward the corresponding events A' to a query deployment that is optimized for detecting (A,B1,D2) (see operator 3 in Figure 3, right side). This operator matches incoming events A,B1,D2 and forwards them to the sink. Incoming D2 without preceding events A,B1 are forwarded to the sink as well. This is because we know from the process model that A,B2 must have occurred before.

Figure 3 illustrates the effect on query deployment in the network. The figure visualizes event sources, locations for query operators, and the query sink in

a latency space. The arrows mark the data flow. The thickness of the arrows roughly corresponds to the consumed bandwidth. Dotted arrows mark event streams where a portion is filtered out by the operators. An efficient deployment decentralizes processing and minimizes the product of bandwidth and latency. Figure 3 illustrates that high bandwidth event streams travel shorter distances when using process knowledge for optimization. This reduces the overall network load in the system and thereby the efficiency of the query processing. Another advantage is that the optimized deployment distributes load more evenly across resources than the original one. Note, that each operator must handle fewer events in the optimized deployment.

5 Conclusions and Future Work

In this paper we presented optimizations for in-network processing based on process knowledge. We illustrated how to exploit timing information and decision points. In future work we plan to quantify the effect of our optimizations with simulations and experiments in distributed computer network PlanetLab.

References

1. EPCglobal. Epc information services (epcis) version 1.0.1 specification (2007), `http://www.epcglobalinc.org/standards/epcis/epcis_1_0_1-standard-20070921.pdf` (last accessed 10/6/2009)
2. Fiege, L., Muhl, G., Pietzuch, P.R.: Distributed Event-based Systems. Springer, Heidelberg (2006)
3. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., Boston (2001)
4. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tinydb: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. 30(1), 122–173 (2005)
5. Owen, S.D., Cooper, O., Edakkunni, A., Franklin, M.J., Hong, W., Jeffery, S.R., Krishnamurthy, S., Reiss, F., Rizvi, S., Wu, E.: Hifi: A unified architecture for high fan-in systems. In: VLDB, pp. 1357–1360. Demo (2004)
6. Patankar, A., Segev, A.: An architecture and construction of a business event manager. In: Etzion, O., Jajodia, S., Sripada, S. (eds.) Dagstuhl Seminar 1997. LNCS, vol. 1399, pp. 257–280. Springer, Heidelberg (1998)
7. Pietzuch, P., Ledlie, J., Shneidman, J., Roussopoulos, M., Welsh, M., Seltzer, M.: Network-aware operator placement for stream-processing systems. In: ICDE 2006, Washington, DC, USA, p. 49. IEEE Computer Society, Los Alamitos (2006)
8. Srivastava, U., Munagala, K., Widom, J.: Operator placement for in-network stream query processing. In: PODS 2005, pp. 250–258. ACM Press, New York (2005)