Mastering Real-Time Big Data With Stream Processing Chains

To conciliate application logic concerns with event handling performance, we introduce the spChains processing framework.

By *Dario Bonino and Luigi De Russis*

DOI: 10.1145/2331042.2331050

Pervasive applications rely on increasingly complex streams of sensor data continuously captured from the physical world. The high granularity and frequency of captured data, however, is seldom compatible with application-level logic and requirements. To overcome this issues we introduce the spChains processing framework, based on state-of-the-art complex event processing (CEP) engines, specifically designed to support flexible, and reusable, monitoring and alerting processes for hundreds of sensors deployed in multiple locations. While maintaining single-event handling capabilities, even on large

settings, spChains allows easy computation of aggregate measures by means of reusable and modular stream processing blocks.

Pervasive applications are increasingly relying on complex and distributed monitoring infrastructures, to timely capture fresh data about the current context and to infer the right actions to perform, be they fully automatic or involving some user decisions. However, the amount of new measures and environment parameters imposes stricter requirements on monitoring networks and software, as both single-event granularity and aggregate measures must be successfully tackled. The former involves treating high-throughput/high-dimensional data streams; for example, 1,000 sensors sampled at 1Hz frequency (typical sampling frequency for residential-class smart environments) require an event han-

dling throughput of 1,000 events per second. The latter requires high-level primitives to define effective data processing, aggregation, and summarization, typically involving data delivery rates depending on the application concerns, e.g., one sample every minute for moderately complex inferences.

Complex event processing [1, 2, 3, 4] initially developed in the context of business process management and operations research [5, 6], is currently assuming a relevant role in data-centric elaboration, offering reliable, fast and cost-effective solutions for handling high throughput data streams. Typical CEP engines effectively handle data at rates between 1,000 to 100,000 messages per second, by relying on a number of techniques involving event-pattern detection, event abstraction, event hierarchies, and so on [7, 8]. CEP has reached a rather high maturity with several tools al-

ready available, ready to be integrated as data processing layers. However its application is still confined to a niche of stream-processing experts as quite deep knowledge of the inner processing mechanisms and engines is required to effectively setup usable monitoring functions (see Jeffery et al. and Wang et al. for a glimpse of required skills [9, 10]). While easily responding to high throughput/high-dimensional data elaboration, at single event granularity, current CEP engines are difficult to adopt in pervasive applications, as involved CEP syntaxes for defining data aggregation and delivery chains are typically complex and bounded with specific CEP implementations, preventing effective design and reuse of data stream computations.

To tackle single-event granularity, high throughput, elaboration while keeping data aggregation design ac-

Figure 1. The spChains logic architecture.

cessible to non-expert users (e.g., the pervasive environment design team) we introduce a data-centric processing framework—spChains [11]—based on state-of-the-art CEP engines and specifically designed to decouple low-level CEP query writing from high-level definition of monitoring processes, which can be carried by teams having reduced knowledge of CEP systems.

We demonstrate the flexibility and effectiveness of the spChains framework using a two-phase experimentation process. In the first phase, performance characterization is carried, showing that the current spChains implementation can easily handle from 6,000 to 170,000 events per second, depending on the required processing. Secondly, a real-world trial on some commercial applications is analyzed and results confirm the flexibility of the approach and its applicability in typical enterprise-level settings.

UNDERSTANDING SPCHAINS

spChains represents monitoring (and alerting) tasks in form of reusable and modular processing chains built atop of a set of 14 standard, and extensible, stream processing blocks. (Figure 1 reports the corresponding logic architecture.) Each stream processing block encapsulates a single (parameterized) stream query, e.g., windowed average or threshold checking, and can be cascaded to other blocks to obtain complex elaboration chains (See Figure 2), using a pipes-and-filter composition pattern [12].

spChains blocks do not represent the complete set of elaborations al-

lowed by full CEP engines; instead they are focused on providing a flexible, reusable and easy-to-learn processing facility for non-experts, with a particular focus on needs emerging from the energy and context monitoring domains, while retaining CEP optimization in single block implementations.

Both aggregation (See Figure 2a) and alerting (See Figure 2b, c) are supported through two main families of processing blocks respectively handling Boolean or real events. Blocks can be mixed and connected according to well-defined composition semantics.

IMPLEMENTATION

The spChains framework is implemented as an open source Java library (<http://elite.polito.it/spchains>), distributed with an Apache v2.0 license. It provides an abstract implementation of the logical modules of the architecture together with all the utilities needed to automatically verify and establish block connections. The base library is composed of 14 blocks implemented by exploiting a state-of-the-art CEP engine called Esper [8]. The Esper-based implementation of spChains supports effective handling of up to 170,000

Figure 2. Stream processing chains for (a) aggregation and (b, c) alerting.

Aggregation

(a) Average on 1h chain composed by a single average block, using a batch temporal window.

Alerting

(b) Detection of sudden temperature changes, provides alerts for variations greater than 0.1°C in the last 10s.

(c) Detects if the sum of pieces produced by a machine is out of the operative range [1,000–5,000 pieces/hour]

events per second. The base block library, as well as any new block implementation, is discovered at run-time by exploiting the Java Services pattern [13], which allows for easily extending the core spChains.

EXPERIMENTAL RESULTS

spChains underwent a two-level testing process aimed at assessing its applicability in possibly large real-world scenarios. To perform this assessment two separate test phases have been deployed: A first in-lab performance test aimed at assessing the performance of each block, in terms of achieved throughput and memory occupation, and a second real-world deployment of the spChains framework in three applications developed by third parties and focused on the energy monitoring domain.

Phase 1: Performance characterization. The 14 blocks composing the spChains base library have been tested, by connecting them with a random event generator and increasing the event delivery rate of the generator from 100 events per second up 170,000 events/s, with an increase factor of two (i.e., doubling the event generation rate at each step). Adopted metrics include the achieved throughput (in healthy work conditions, i.e., without losing events in output), compared to the theoretical one, and the corresponding memory occupation. Both have been averaged more than 10 trials for each of the adopted generation rates.

Analyzed blocks have been categorized in three different typologies:

spChains represents monitoring (and alerting) tasks in form of reusable and modular processing chains built atop of a set of 14 standard, and extensible, stream processing blocks.

aggregation blocks, blocks without memory, and pattern-based blocks. *Aggregation blocks* compute aggregated measures such as average, maximum, or minimum, on a user-defined time window. They typically provide a constant output rate (one output event every time window expiration), with a memory occupation, which increases almost linearly with respect to the rate of input events. *Blocks without memory* typically involve single event operations such as Abs or Scale, without event pattern matching. Since these blocks operate on single events, they are more sensitive to the computation performance of the implemented CEP queries. Pattern-based blocks implement computations based on pattern matching, e.g., the And, Delta, and Difference blocks. As reported in Table 1, they

offer the worst performance of the three block groups, achieving a maximum throughput of only 6,000 events per second.

Phase 2: Real-world trial. spChains has been tested for real-world applicability in two technology transfer projects involving Italian companies located in Turin and Milan, respectively. In both cases, spChains has been integrated into energy monitoring applications: Enterprise Resource Planning software in the former case (one instance) and energy managers’ dashboards in the latter (two installed instances). Events generated by field sensors—38 in the first case and 47 in the second installation—were monitored 24/7 with a sampling period of 1 event per second (per sensor) and with a typical chain delivery time of 15 minutes (aggregated measures use 15 minutes windows, typically). A formal assessment of spChains reliability has not yet been devised; however preliminary, qualitative feedback has been gathered from the two companies and shows a rather high rating of the system in terms of performance and up-time.

CONCLUSIONS

We introduced the spChains framework, a modular approach to master complex event processing queries in a simplified, yet effective, manner based on stream processing block composition. While trading off global query optimization with modular composition, the spChains framework is able to handle high-cardinality/high-throughput data flows, with peak processing performance at around 170,000 events per second.

Exploitation in the real-world is sustainable, as shown by two commercial applications (in three different installations). Currently, we are collaborating with the energy managers of the Politecnico di Torino to deploy a University-wide monitoring network comprising more than 300 sensors. Future works will involve further optimization/integration of the base block library, the development of chain definition interfaces (standalone and Web-based), and integration of chain/block hot-plugging functionalities.

Table 1. Block performance analysis.

Block Typology	Block Name	Input Rate	Throughput	Memory Occupation
Aggregation Blocks	Sum	150k event/s	N/A	300 MByte
	Avg	140k event/s	N/A	275 MByte
	Last	170k event/s	N/A	300 MByte
	Max	120k event/s	N/A	170 MByte
Blocks without memory	Abs	N/A	42k event/s	100 MByte
	Scale	N/A	42k event/s	100 MByte
	Join	N/A	42k event/s	100 MByte
Pattern-based blocks	And	6k event/s	6k event/s	<20MByte
	Delta	6k event/s	6k event/s	<20MByte

References

- [1] Luckham, D. C. *The Power of Events: An introduction to complex event processing in distributed enterprise systems*. Addison-Wesley Longman Publishing Co., Boston, 2001.
- [2] Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., and Zdonik, S. Aurora: A new model and architecture for data stream management. *The VLDB Journal* 12, 1 (August 2003), 120–139.
- [3] The Stream Group. Stream: The Stanford stream data manager. Technical Report 2003-21. Stanford InfoLab, 2003; <http://ilpubs.stanford.edu:8090/583/>
- [4] Etzion, O., and Niblett, P. *Event Processing In Action*. Manning Publications, Greenwich, CT, 2010.
- [5] Kellner, I., and Fiege, L. Viewpoints in complex event processing: Industrial experience report. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems (DEBS '09)*. ACM Press, New York, 2009, 9:1–9:8.
- [6] Zang, C., and Fan, Y. Complex event processing in enterprise information systems based on RFID. *Enterprise Information Systems* 1, 1 (2007), 3–23.
- [7] Dindar, N., Güç, B., Lau, P., Ozal, A., Soner, M., and Tatbul, N. Dejavu: Declarative pattern matching over live and archived streams of events. In *Proceedings of the 35th SIGMOD international conference on Management of data (SIGMOD '09)*. ACM Press, New York, 2009, 1023–1026.
- [8] Oberoi, S. Esper complex event processing engine. *Embedded System Engineering* 16, 2 (2011), 28–29.
- [9] Jeffery, S. R., Alonso, G., Franklin, M. J., Hong, W., and Widom, J. Declarative support for sensor data cleaning. In *Proceedings of the Fourth International Conference on Pervasive Computing (PERVASIVE'06)* Springer-Verlag, Berlin, 2006, 83–100.
- [10] Wang, W., Sung, J., and Kim, D. Complex event processing in EPC sensor network middleware for both RFID and WSN. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium*. 2008, 165 –169.
- [11] Bonino, D. and Corno, F. spChains: A declarative framework for data stream processing in pervasive applications. In ANT 2012, The Third International Conference on Ambient Systems, Networks and Technologies, 2012.
- [12] Garlan, D. and Shaw, M. An Introduction to Software Architectures. Carnegie Mellon University, Tech. Rep. CMU-CS-94-166, January 1994.
- [13] O'Conner, J. Creating Extensible Applications With the Java Platform. Oracle, Java, Tech. Rep., 2007.

Biographies

Dario Bonino, Ph.D., is a research assistant with the e-Lite research group in the Department of Computer Science and Automation of Politecnico di Torino. He received his M.S. and Ph.D. degrees in electronics and computer science engineering, respectively, from Politecnico di Torino in 2002 and 2006. His current research interests include semantic Web technologies, with a particular focus on architectures for semantic annotation, indexing and retrieval of Web resources, domotics and semantic-aware, home-related technologies, distributed systems, multi-agent systems and semantics-aware Web architectures.

Luigi De Russis is a Ph.D. student at the Department of Computer Science and Automation of Politecnico di Torino. He received his M.S. degree in computer engineering from Politecnico di Torino in 2010. His current research focuses on human computer interaction, with a particular interest on distributed user interfaces and interaction techniques applied to smart environments and intelligent domotic environments.