BiCEP
Benchmarking Complex Event Processing Systems

Pedro Bizarro

University of Coimbra, DEI-CISUC
3030-290 Coimbra, Portugal
bizarro@dei.uc.pt

Abstract. BiCEP is a new project being started at the University of Coimbra to benchmark Complex Event Processing systems (CEP). Although BiCEP is still in the early stages, we list here some of the design considerations that will drive our future work and some of the metrics we plan to include in the benchmark.

Keywords: Complex Event Processing, benchmark, synthetic benchmark, events, response time, throughput, scalability, adaptivity, query processing

1 Introduction

A Complex Event Processing (CEP) system is a relatively new kind of software system that is being cast as a fundamental central piece in a variety of scenarios that deal with the detection of complex patterns within series of in-coming events (or tuples or messages). Typically, the events come from many distributed sources, at very high rates and possibly out-of-order. While it processes the in-flight events, the CEP engine may have to analyze large amounts of historical data to be able to detect the patterns of interest [10]. Examples of complex events are: atypical heart sequences, interesting stock value patterns (e.g, the triple-bottom pattern [13]), computer network intrusion, suspicious credit-card purchases, unsafe airplane flight paths. Frequently, detecting an event represents very valuable information and requires immediate follow-up action by another system or by a human. As such, applications may require very short response times (e.g., sub-second or less) from the moment the events arrive to the system until the CEP engine notifies the complex event detection.

However, although CEP applications share common requirements, there is wide variability in many others: some require 2D and 3D space operations, others require the probabilistic prediction of future events, some require transactional guarantees, some need millisecond response times while others can accept minute or hour response times, some deal with just a few events per second while others analyze millions or billions of events per second.

In spite of the lack of common CEP requirements, many existing software companies, new startups, open-source groups and research groups are developing their own CEP engines. Many of these new CEP engines have different architectural heritages or inspirations: some are derived from rule-engines, others from data stream management systems, still others from active or temporal database research, others from messaging and queuing systems, while still others are being built from scratch. These architectures seem to be too far apart: currently there are no agreed upon terminology, semantic, query language, data formats, APIs, standards, or benchmarks in the CEP community.

From a research point of view, the lack of common models, semantics, standards and benchmarks makes the comparison of the existing approaches hard and hinders the development of new algorithms and solutions.

The goal of the BiCEP project is to identify some of the core CEP requirements and develop a synthetic benchmark (or possibly a set of benchmarks) to allow a comparison of products and algorithms in spite of their architectural and semantic differences. In the next two sections we describe some of the design considerations and metrics we are planning to include in the benchmark.

2 Benchmark Design Considerations

A good benchmark should be relevant (or representative), portable, scalable, and simple [8]. To be representative, a synthetic benchmark like BiCEP needs to mimic the typical characteristics of the applications it models. In the 1st and 2nd Event Processing Symposiums [4, 5], at the Dagstuhl Event Processing Seminar [6], and at the International Conference on Distributed Event-Based Systems in June 2007 [9] many use cases have been presented. Most fall into the following categories:

- Business activity monitoring
- RFID applications
- Event processing in workflows
- Risk and compliance applications
- Financial trading
- Health monitoring
- Telecommunications
- Military
- Transportation and assignment
- Scientific computations
- Intrusion and fraud detection

Although all users would prefer CEP engines that can cope with high event throughputs and have shorter response times, applications for the different domains above have very different performance requirements. Response time requirements can range from milliseconds (e.g., for financial trading) to seconds (e.g., health monitoring) to minutes or more (e.g., fraud detection). Likewise, throughput can range from a hundreds of events per second (e.g., in health monitoring) to billions of events per second (e.g., scientific computations). It is unlikely that a single-domain synthetic benchmark is able to explore this wide range of performance requirement scales. We expect that, even if delivered as a single benchmark, BiCEP will turn out to be a set of small single-domain synthetic benchmarks with different data sets and different queries.

The variety of CEP domains and the lack of standards in query languages and data formats poses more benchmark design challenges. For example, unlike the exact semantic meaning of SQL queries in TPC benchmarks [14], BiCEP will describe queries in a meta-language or in English. Likewise, events may be produced in simple formats (e.g., XML) and it will be left as a responsibility of the CEP engine being benchmarked the transformation of the incoming events into its own data format and the transformation of output events back into the benchmark format.

To properly measure all the event processing activity by the CEP engine system, BiCEP will be designed to interface with the CEP engine as shown in Figure 1: BiCEP will produce all the input to, and consume the result output from the CEP engine being benchmarked. This model will guarantee that any buffering, event cleaning or event transformation activity that happens at the CEP engine is taken into account in the overall performance numbers. Given the lack of query language standards, it is not clear yet which system should orchestrate query generation (shown as a dotted rectangle in Figure 1).

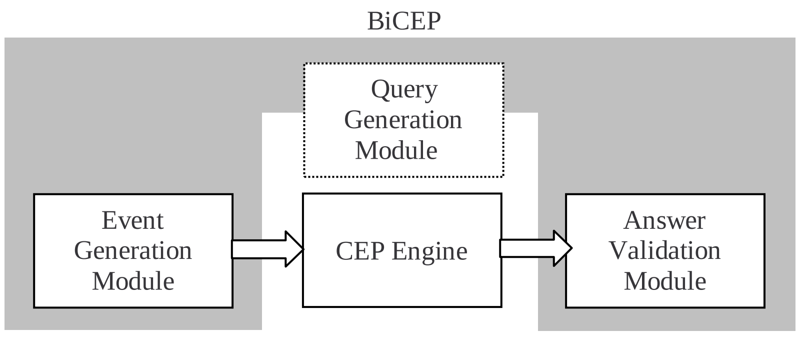

Figure 1. The interface between BiCEP modules and the CEP engine

The benefits of synthetic benchmarks are well understood: data availability, experimental control, and scalability. The main challenge will be to develop a synthetic benchmark that is representative of such a wide range of CEP applications and at the same time is simple to understand and is widely accepted by users, developers, and researchers.

3 Benchmark Metrics

In this section we describe some of the metrics we believe a CEP benchmark should assess:

- **Sustainable throughput**: the steady-state number of events per unit of time that a (warmed-up) CEP engine can process while performing query processing. Even within the same system, sustainable throughput can vary widely depending on the amount of work to be done during query processing.

- **Response time**: the time since the last event of some event pattern is fed into the system until the system notifies the event pattern detection.
- **Scalability**: Unlike other benchmarks that considerer scalability only as a variation of the benchmark with more data and more users, in BiCEP we would like scalability to be a first-class metric. That is, while it is useful to compare systems at different scale levels, it is also very interesting to assess how well a given system scales. Thus, we plan to devise experiments that, e.g., significantly increase the load (events and queries) and measure how well the CEP engine copes with the load-up. The development of new techniques (e.g., virtualization techniques used by Amazon Elastic Compute Cloud [1] and Enomalism Elastic Computing [3]) that allow a system to grab hardware resources on demand makes, in our opinion, scalability experiments especially interesting. We are planning scalability experiments along three directions: i) **scale-up:** increase the system and increase the load, ii) **speed-up:** increase the system and maintain the load, and iii) **load-up:** maintain system but increase the load.
- **Adaptivity**: Typically, systems are benchmarked after they are "warmed-up" and in a steady state. However, while it seems that there will be periods where CEP systems are in steady states, it also appears likely that, due to the very unpredictable nature of the real-world events being processed by CEP engines, there will be frequent disruptive moments, when the system should adapt its query processing to be more efficient. We are planning a series of experiments that measure how well a CEP system copes with sources of change. Some of these planned experiments include: changing the event/query arrival rate from a steady state to different steady state, insert a short bursty arrival of events, produce a long system overload, and introduce disconnects, network delays, and buffering.
- **Computation Sharing**: Many CEP applications process tens, hundreds, millions of similar queries concurrently. For example, a CEP engine in a financial trading company may be processing thousands of rules for each stock ticket: many customers may be monitoring the same stock but each customer may have slightly different buy or sell values. If the CEP engine can devise query processing techniques such that different queries are able to share computation, then the scalability potential of the system is greatly improved. We plan to have scenarios in the benchmark that test this situation.
- **Similarity search and precision and recall**: As far as we know, no CEP engine uses any kind of similarity search: the patterns being searched are always precisely specified by a query language. Thus, we expect no false positives and no false negatives. However, if CEP users demand more and more complex patterns, we expect CEP engines to start using similarity search. If similarity search is used, then CEP engines may occasionally produce incorrect results by way of false positives and false negatives. We also expect false positives and false negatives if CEP engines use past events to forecast real-world future events. Although we do not plan to include them in the first BiCEP version, we expect that soon CEP benchmarks will include information retrieval metrics (e.g., precision and recall [11]) in addition to the other performance oriented metrics.

4 Other Issues Affecting Performance

Besides the primary benchmark metrics listed above, there are a number of other issues that will likely affect performance significantly. Some of these issues led us to raise the following questions:

- Will the CEP engine provide transactional guarantees? If yes, is the user allowed to trade some of the performance guarantees switching them off in exchange for performance improvements? What is the impact of coupling modes on performance? (Coupling modes determine when the Action part of an ECA rule is fired: immediately upon event detection, before transaction commit, or in a separate transaction [7].)
- What is the impact of out-of-order events in performance? How much out-of-order can the events be without affecting throughput?
- What is the impact of event record size on performance? Will the system behave as well with a 2-field event record as with a 200-field event record?
- What is the impact of query complexity on, say, maximum throughput? What is the maximum throughput for simple event aggregation and counting? What is the maximum throughput for very complex pattern and correlation detection rules?
- When comparing current events with past history, how far back in the past can the CEP engine look up without visible performance impact?
- Are all the queries the same priority? For example, if a query in a military CEP scenario is searching for a “incoming missile: 10-second to impact” pattern and at the same time the CEP engine is collecting and aggregation temperature values from battlefield sensors, what query should have processing priority when the system is overloaded? Will there be ways to specify query priority? Will the priority scheduler work as expected?
- What types of communication (push, pull, scheduled) are allowed and what is the impact on performance, and what queries benefit from which mechanism?

5 Summary and Related Work

There are at least three other benchmarks that are relevant to CEP and that we will use as partial inspiration and guidelines when designing BiCEP: the BEAST benchmark for Object-Oriented Active Database Systems [7], the Linear Road benchmark for Data Stream Systems [2], and the soon to be released, SPECjms2007 benchmark for Message-Oriented Middleware [12]. Although any of these benchmarks measures activities that BiCEP will also measure (e.g., event detection, window-based tuple aggregation, and the underlying message passing and queuing layer), BiCEP will more comprehensively focus on modern CEP engines, applications, and requirements. Specifically, our project’s main goal is to identify the core CEP requirements and develop a benchmark to compare and assess the merits of CEP products and algorithms in spite of their architectural and semantic differences.

BiCEP will be designed to measure sustainable throughput, response time, scalability, adaptivity, and the ability to share computation between different queries.

BiCEP tests will include a variety of scenarios considering different transactional properties, different levels of pattern complexity, in order and out-of-order events, different history lookup windows and different communication mechanisms.

The most recent BiCEP developments can be found from the project's web page at: http://bicep.dei.uc.pt.

Acknowledgements

The BiCEP project is a 3-year project funded by a European Commission FP6 Marie Curie International Reintegration Grant. The project officially starts June 2007.

References

[1] Amazon Elastic Compute Cloud – Amazon EC2. Available at http://www.amazon.com/.

[2] Arvind Arasu, Mitch Cherniack, Eduardo F. Galvez, David Maier, Anurag Maskey, Esther Ryvkina, Michael Stonebraker, Richard Tibbetts. Linear Road: A Stream Data Management Benchmark. In *Proc. of the Intl. Conf. on Very Large Data Bases (VLDB'2004)*, Toronto, Canada. August 2004.

[3] Enomalism Elastic Computing. Available at http://enomalism.com/.

[4] Opher Etzion (General Chair). *1st Symposium on Event Processing*. IBM Research, Yorktown Heights, NY. March 2006. Presentations available at http://complexevents.com/?page_id=87.

[5] Opher Etzion (General Chair). *2nd Symposium on Event Processing*. Oracle, San Mateo, CA, USA. November 2006. Presentations available at http://www.complexevents.com/?page_id=129.

[6] Opher Etzion (General Chair). *Dagstuhl Event Processing Seminar*. Dagstuhl, Germany. May 2007. http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=2007191.

[7] Andreas Geppert, Mikael Berndtsson, Daniel Lieuwen, Claudia Roncancio. Evaluation of Active Database Management Systems Using the BEAST Benchmark. *Theory and Practice of Object Systems*, Vol. 4(3), 135–149 1998.

[8] Jim Gray (Editor). *The Benchmark Handbook for Database and Transaction Processing Systems*. Morgan Kaufmann. 1993.

[9] Hans-Arno Jacobsen (General Chair). *The Inaugural Intl. Conf. on Distributed Event-Based Systems (DEBS2007)*, Toronto, Canada. June 2007.

[10] David Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Professional; 1st edition. May 8, 2002.

[11] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1st edition, June 1999.

[12] Kai Sachs, Samuel Kounev, Jean Bacon, and Alejandro Buchmann. *Workload Characterization of the SPECjms2007 Benchmark*. To appear.

[13] The Stock Bandit. *Triple Bottom Pattern*. Available at http://www.thestockbandit.com/Triple-bottom.htm, accessed July 2007.

[14] Transaction Processing Performance Council. http://www.tpc.org/.