Quality Analysis and Verification of Data-Intensive Applications

Francesco Marconi

DEIB, Politecnico di Milano, Milan, Italy
francesco.marconi@polimi.it

Abstract Data-intensive applications are gaining momentum in multiple domains of science and business, thanks to the technological advancements of distributed systems and the increased awareness of the importance of data. However, the fast-paced development of such technologies is followed by a lack of methodological support for the design of such applications. Quality requirements regarding safety and data privacy are often overlooked by designers, increasing the number and severity of quality-related failures, imposing heavy refactoring of the applications. This work aims at providing a methodology and tool enabling the formal analysis and verification of quality properties in data-intensive application. The preliminary stage of this work is focused on safety analysis of topology-based streaming applications, but the approach could be extended to other technologies and encompass a wider range of properties.

Keywords: Data-intensive Applications, Distributed Systems, Formal Verification, Metric Temporal Logic, Apache Storm

1 Introduction

Big Data technologies have shown significant development and diffusion in recent years. The so called *"data-intensive" applications* [6], or *DIAs*, exploit the increasing maturity and accessibility of emerging technologies (such as cloud computing, MapReduce/Hadoop, NoSQL databases, stream processing) in order to process massive quantities of data in a timely manner. Since these technologies are inherently complex and lack of standard development methodologies, both academia and industry are involved in researching innovative solutions to support the entire life-cycle of DIAs, with particular attention to quality requirements.

The DICE project [11] defines techniques and tools for the data-aware quality-driven development of DIAs. It offers tools for quality assessment, architecture enhancement, agile delivery and continuous testing of DIAs. As contributors to the DICE project we are investigating the usage of formal methods to perform safety and privacy analysis on the design of DIAs.

The research question addressed by this work is:
"How can we verify quality properties, such as safety and privacy, of data-intensive applications?".

This question, having such a broad scope, triggers other challenges, such as: *(i)* Which (application-dependent) properties could we verify? Associated to which technologies? *(ii)* How can we devise a mechanism to automatically carry out formal verification, allowing the designer (i.e., not a formal methods expert) to define the application without having to directly deal with the details of the formal model?

Before describing our approach, it is appropriate to remark that there are multiple dimensions to take into consideration in order to perform formal analysis and verification on DIAs. These dimensions include: *(i) functional* aspects of the application: DIAs are typically composed of multiple distributed and multi-threaded components, each of them performing some processing on data. Relevant features, such as operations performed, input and output data, can be considered (and captured in the model) with different granularity. Depending on the goal of the analysis, it might be relevant to represent the operations at the thread level, at the component level or even at a coarser level of granularity. Details add complexity to the model, impacting the computation time to run verification, so a trade-off is often needed. *(ii) non-functional* aspects are, by definition, of primary importance when talking about quality analysis. Different features can be targeted, such as performance, availability, safety, privacy, etc. In our case all the relevant non-functional aspects are related to time. *(iii) infrastructural* aspects need to be considered in DIAs: applications typically run on distributed systems that add a further layer of complexity to the analysis. Also in this case, the appropriate choice of the abstraction level is crucial. Focusing on one or more of these dimensions will determine the kind of analysis to undertake.

2 Approach

Given the wide range of technologies and frameworks available in the Big Data ecosystem, we decided to focus on a specific set of applications (i.e., streaming applications), analyze it and identify related safety issues. We then devised a formal model capturing the behavior of those applications, and allowing for the representation of such issues as violations of safety properties.

Many of these applications can be described by means of *topologies*, that is, directed graphs of computation. Topologies are composed of two kinds of nodes: *computational nodes*, representing the logical operations performed over data, and *source nodes*, representing the entry point of data into the application. Topologies are implemented differently across the various frameworks, but are still able to represent, with a certain degree of abstraction, most of the streaming applications. The work currently focuses on the Apache Storm [1] technology, a well known streaming framework that allows for parallel, distributed, real-time processing of large-scale streaming data on horizontally scalable systems.

In Storm computational nodes are called *bolts*, and source nodes are called *spouts* (see Fig. 1). All the nodes have dedicated input queues from which they read data to be processed. One of the key concerns in Storm applications is that time-related parameters, such as emission rates of data, may induce an

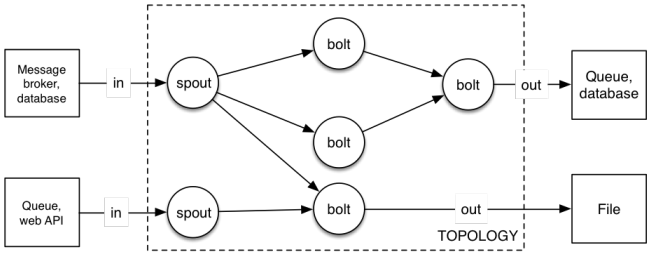

Figure 1: Example of Storm topology.

excessive load on the topology by accumulating data in bolts' queues. Since streaming applications are supposed to be permanently active, and the streams of data consist in potentially infinite sequences of messages, the occurrence of such overloads may lead to serious degradation of crucial quality aspects like latency and throughput. For this reason we decided to capture such concern as an unwanted behavior of the system against which the designer wants to analyze the safety of the design.

We devised novel a model called *Timed Counter Networks* (TCN) [9], in which we modeled the behavior of spouts and bolts as parametric components that can be used as building blocks for defining topologies. The parameters characterizing each component are essentially: the parallelism level (number of threads executing its operations) for both spouts and bolts, the processing time of single messages and the kind of function performed (filter, transformation, join of multiple streams) for bolts, the average number of messages produced per time unit for spouts. The model is expressed through *Constraint LTL over-clocks* (CLTLoc) [4] metric temporal logic enriched with positive counters. The choice of the formalism is motivated by its flexibility and by the fact that it is supported (provided some limited extensions) by the Zot[1] bounded model/satisfiability checker. This allowed us to define a tool supported mechanism [3] for automating the verification.

The tool, named D-VerT[2], allows for the definition of topologies as UML Class diagrams, conveniently annotated with a specific UML profile, and performs a series of transformations to create the corresponding formal model and to run verification on it (see Fig. 2). The verification outcome is presented to the user both in a textual and in a graphical format, in order to provide a visual hint about the potential design issues. The UML modeling, together with the automatic transformation mechanism and the graphical display of results aim at hiding the complexity of the underlying models and engines to the user, minimizing the need of technical expertise in formal methods.

In the current stage of development the tool supports verification of the property concerning the aforementioned unbounded growth of load on bolts'

[1] https://github.com/fm-polimi/zot
[2] https://github.com/dice-project/DICE-Verification

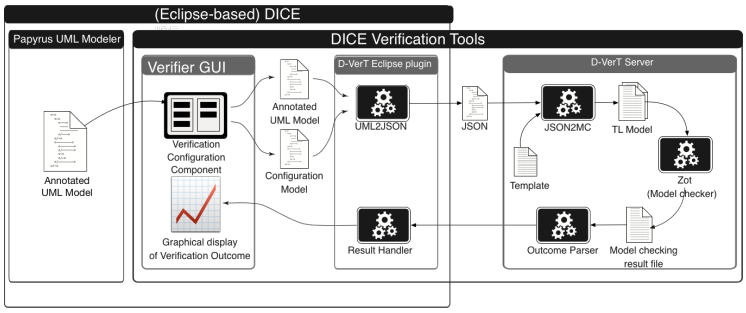

Figure 2: Architecture and execution flow of D-VerT

queues, i. e., *"all bolts' queues have a bounded number of elements"*. We argue that, if this property holds, then all bolts are able to process the incoming flow of data in a timely manner. The property is disproved if there exists an execution where at least one of the queues grows with an unbounded trend. This kind of analysis is enabled by adapting the Bounded Satisfiability Checking (BSC) technique used for CLTLoc [2,5] to deal with the addition of discrete counters that makes Timed Counter Networks undecidable. More details about the verification technique can be found in [9].

The preliminary version of D-VerT is available either as part of the DICE Platform[3] or as a standalone application. As shown in Fig. 2, its client-server design includes a client component, implemented as an Eclipse plug-in and a RESTful web server component, distributed as a multi-container application.

3 Preliminary and expected results

We ran experiments on different topologies in order to have qualitative feedback about the analysis performed by D-VerT and to collect quantitative measures of the verification time. Some of the preliminary outcomes of such experiments are presented in [9]. Under the qualitative point of view, the tool was able to spot some meaningful design flaws on use-case topologies provided by industrial partners. Execution times have shown a relevant dependency on the number of nodes composing the topology, but also on the specific configuration of each node (i. e., some topologies with evident design flaws took considerably less time to be analyzed with respect to other nontrivial ones).

Future works will be conducted on several directions. Our plans include: *(i)* mitigating the state-space explosion problem registered when the model includes too many components; *(ii)* investigating new technologies, such as Apache Spark, Flink or Tez; *(iii)* reasoning about new properties to verify; *(iv)* examining in depth the theoretical aspects of the timed counter networks model in order to get new results in terms of soundness and completeness.

[3] https://github.com/dice-project/DICE-Platform

4 Related works

Various approaches exist for the formalization of distributed systems; however, to the best of our knowledge, none focuses on topology-based DIAs. In the context of DIAs, some works aim at verifying properties that are related to the framework itself such as reliability and load balancing [12], or the validity of messaging flow in MapReduce [13]. Our work is instead focused on supporting "application-dependent" analysis. There are similarities between the analysis proposed in our work and some aspects of the analysis of networks, as they both deal with properties related to throughput and latency in topology-based systems. However, the latter generally focuses on different kind of problems, such as the verification of routing protocols [10].

Timed counter networks are inspired from *Vector Addition Systems with States* (VASS) [8] and Timed Petri Nets [7].

Acknowledgment This is a joint work with Marcello M. Bersani under the supervision of Matteo G. Rossi, supported by Horizon 2020 project no. 644869 (DICE).

References

1. Apache Storm, http://storm.apache.org/
2. Bersani, M.M., Frigeri, A., Morzenti, A., Pradella, M., Rossi, M., Pietro, P.S.: Constraint LTL satisfiability checking without automata. J. Applied Logic 12(4), 522–557 (2014)
3. Bersani, M.M., Marconi, F., Rossi, M., Erascu, M.: A tool for verification of big-data applications. In: Proc. of QUDOS. pp. 44–45 (2016)
4. Bersani, M.M., Rossi, M., San Pietro, P.: A tool for deciding the satisfiability of continuous-time metric temporal logic. In: Proc. of TIME. pp. 99–106 (2013)
5. Bersani, M.M., Rossi, M., San Pietro, P.: A tool for deciding the satisfiability of continuous-time metric temporal logic. Acta Informatica 53(2), 171–206 (2016)
6. Chen, C.P., Zhang, C.Y.: Data-intensive applications, challenges, techniques and technologies: A survey on big data. Information Sciences 275, 314–347 (2014)
7. Furia, C.A., Mandrioli, D., Morzenti, A., Rossi, M.: Modeling Time in Computing. Monographs in Theoretical Computer Science. An EATCS Series, Springer (2012)
8. Karp, R.M., Miller, R.E.: Parallel program schemata. Journal of Computer and System Sciences 3(2), 147 – 195 (1969)
9. Marconi, F., Bersani, M., Erascu, M., Rossi, M.: Towards the formal verification of data-intensive applications through metric temporal logic. In: Proc. of ICFEM 2016. LNCS, vol. 10009, pp. 1–17 (2016), to appear
10. Qadir, J., Hasan, O.: Applying formal methods to networking: Theory, techniques, and applications. IEEE Communications Surveys & Tutorials 17(1), 256–291 (2015)
11. The DICE Consortium: DICE Verification Repository (Jan, 2016), URL: https://github.com/dice-project/DICE-Verification
12. Tommaso Di Noia, M.M., Sciascio, E.D.: A computational model for mapreduce job flow (2014)
13. Yang, F., Su, W., Zhu, H., Li, Q.: Formalizing mapreduce with csp. In: Proceedings of ECBS. pp. 358–367. IEEE Computer Society (2010)