Automated Architecture Recovery and Consistency Checking for Stream-Based Systems

Marcello Bersani, Francesco Marconi and Damian A. Tamburri
Politecnico di Milano
Milan, Italy
Email: [$marcellomaria.bersani, francesco.marconi, damianandrew.tamburri$]@polimi.it

Pooyan Jamshidi, Andrea Nodari
Imperial College London
London, UK
Email: [$p.jamshidi, a.nodari15$]@imperial.ac.uk

***Abstract*—Big data architectures have been gaining momentum in the last few years. For example, Twitter uses complex Stream topologies featuring frameworks like Storm to analyse and learn trending topics from billions of tweets per minute. However, verifying the consistency of said topologies often requires deployment on multi-node clusters and can be expensive as well as time consuming. As an aid to designers and developers evaluating their Stream topologies at design-time, we developed OSTIA, that is, "On-the-fly Static Topology Inference Analysis". OSTIA allows reverse-engineering of Storm topologies so that designers and developers may: (a) use previously existing model-driven verification&validation techniques on elicited models; (b) visualise and evaluate elicited models against consistency checks that would only be available at deployment and run-time. We illustrate the uses and benefits of OSTIA on three real-life industrial case studies.**

I. Introduction

Big data applications process vast amounts of data [1] for the purpose of gaining key business intelligence through complex analytics such as machine-learning [2]. Said applications are receiving increased attention in the last few years given their ability to yield competitive advantage by direct investigation of user needs and trends hidden in the enormous quantities of data produced daily by the average internet user. Gartner predicts[1] that said business intelligence and analytics applications will remain top focus for CIOs until at least 2017-2018. However, there are many costs and complexities behind harnessing said applications ranging from very high infrastructure costs to steep learning curves for the many frameworks (e.g., Apache Storm[2], Apache Spark[3] or Hadoop[4] to name a few) involved in designing and developing applications for Big data.

In our own experience with designing and developing for Big Data, we observed that one such key complexity lies in evaluating the effectiveness of architectures, or better, topologies - in the Big data jargon. In Big data terms, effectiveness means being able to deploy architectures/topologies consistently with restrictions from necessary Big data frameworks (e.g., Spark, Storm, etc.). A very trivial example of said restrictions consists in the need for the topology to represent a Directed-Acyclic-Graph (DAG), a key constraint of the Storm framework. We argue that such effectiveness can be maintained at design time and it does in fact ease expedite deployability of Big data architecture topologies, saving the time and effort of running trial-and-error experiments while setting up expensive infrastructure needs as well as while fine-tuning complex framework setup options.

In an effort to tackle the above complexity, we developed OSTIA, that stands for: "On-the-fly Static Topology Inference Analysis". OSTIA allows designers and developers to infer the application topology though reverse-engineering and architecture recovery [3]. During this inference, OSTIA analyses the inferred topology to make sure it is consistent with restrictions from the necessary Big data frameworks at hand. Currently, OSTIA focuses on Storm, i.e., one of the most famous and established real-time stream processing Big data engine [4]. OSTIA hardcodes intimate knowledge on the development and framework dependence structure necessary for correct Storm topologies for two purposes: (a) realising a visual representation for said topologies; (b) checking said topologies against framework restrictions that would only become evident during infrastructure setup and runtime; (c) finally, export said topologies for further analysis, e.g., through verification techniques [**?**].

This paper outlines OSTIA, elaborating its major usage scenarios and its benefits while properly discussing and addressing its limitations. Also, we evaluate OSTIA on three industrial case-studies from an open-source social-sensing application to show that OSTIA yields valuable design and development insights using an inference analysis of the three static topologies for said application. Finally, we elaborate on how OSTIA helps the quality evaluation of recovered topologies by means of valuable verification techniques. We conclude that OSTIA does in fact provide valuable insights for software engineers to increase quality of their Big data design and development featuring Storm so that expedite deployment can take place.

The rest of the paper is structured as follows. Section II outlines our research problem, research questions and our approach at tackling them. Section III outlines OSTIA, discussing its typical usage scenarios and outlining the main ben-

[1] http://www.gartner.com/newsroom/id/2637615
[2] http://storm.apache.org/
[3] http://spark.apache.org/
[4] https://hadoop.apache.org/

efits we perceived in using it. Section IV elaborates further on the benefits by providing an actual evaluation of OSTIA using three cases from industrial practice. Section V-B discusses the results and evaluation, also outlining OSTIA limitations and potential threats to its validity. Finally, Sections VI and VII outline related work and conclude the paper.

II. Research Design

The work we elaborated in this paper is stemming from the following research question:

"Can we improve the quality and effectiveness of deployable Big-Data streaming designs?"

The material contained in this paper in response to this research question was initially elaborated within a free-form focus group [**?**] involving two experienced practitioners and researchers on streaming technology such as Storm. As a result of the focus group, through self-ethnography [**?**] and brainstorming we designed OSTIA and identified the series of consistency checks that could be run while recovering an architectural representation for Storm topologies. The prototype thus developed was refined incrementally through a series of case-studies. Finally, we applied well established verification approaches to integrate the value perceivable for OSTIA with established benefits stemming form the verification and validation field.

TODO: @anyone, feel free to elaborate more!!

III. Research Solution

IV. Evaluation

- we can use the ATC case study as much as we want - that yields already three topologies that we can infer
- any additional case that we can run?
- what do the results show? do we have a way to quickly quantify the time that is saved by using this approach? e.g., the time that is saved in setting up and running the infrastructure and how much would that time saved have costed these could be valuable evaluation insights

V. Discussion

A. Findings and Quality-Improvement Insights

- we should probably enumerate the many issues we have found with the work in WP2 and WP3 to allow some good insights for the people using OSTIA
- also we could discuss the benefits one by one using some usage-scenarios

B. Approach Limitations and Threats to Validity

- here we could elaborate on the limitations of addressing only storm and why we are addressing storm (e.g., the main technology for streaming currently)
- we should probably discuss the fact that the tech is still eclipse-based and how this is not really a limitation after all
- are there any threats to validity? we should probably discuss this as well

VI. Previous and Related Work

The work behind OSTIA stems from the EU H2020 Project called DICE[5] where we are investigating the use of model-driven facilities to support the design and quality enhancement of Big-Data applications. In the context of DICE, much previous work has been done to support the design, development and deployment of Big-Data applications. For example, we have been developing a series of ad-hoc technological specifications, i.e., meta-model packages that contain all concepts, constructs and constraints needed to develop and operate Big-Data applications for the frameworks coded into the technological specifications. These specifications can be used, for example, to instantiate Big-Data components following standard Model-Driven procedures, without the need to learn said frameworks at all. OSTIA uses insights gained in developing and using said frameworks to apply consistency checks in the context of recovering Big-Data architectures, specifically, for Storm. Much similarly to the DICE effort, the IBM Stream Processing Language (SPL) initiative [5] provides an implementation language specific to programming streams management (e.g., Storm jobs) and related reactive systems based on the Big-Data paradigm. Although SPL is specific to WebSphere and IBM technology, its attempt at providing a relatively abstract language to implement for streams management and processing is remarkably related to OSTIA, since one of its aims is to improve quality of streams management by direct codification of higher order concepts such as streams declarations.

In addition, there are several work close to OSTIA in terms of their foundations and type of support.

First, from a quantitative perspective, much literature discusses quality analyses of Storm topologies, e.g., from a performance [6] or reliability point of view [7]. Said works use complex math-based approaches to evaluating a number of Big data architectures, their structure and general configuration. However, although novel, these approaches do not suggest any significant design improvement method or pattern to make the improvements *deployable*. With OSTIA, we make available a tool that automatically elicits a Storm topology and, while doing so, analyses said topology to evaluate it against a number of consistency checks that make the topology consistent with the framework it was developed for (Storm, in our case). As previously introduced, a very trivial example of said checks consists in evaluating wether the topology is indeed a Directed-Acyclic-Graph (DAG), as per constraints of the Storm framework. To the best of our knowledge, no such tool exists to date.

Second, from a modelling perspective, approaches such as StormGen [8] offer means to develop Storm topologies in a model-driven fashion using a combination of generative techniques based on XText and heavyweight (meta-)modelling, based on EMF, the standard Eclipse Modelling Framework Format. Although the first of its kind, StormGen merely allows the specification of a Storm topology, without applying

[5]http://www.dice-h2020.eu/

any consistency checks or without offering the possibility to *recover* said topology once it has been developed. By means of OSTIA, designers and developers can work hand in hand while refining their Storm topologies, e.g., as a consequence of verification or failed checks through OSTIA. As a consequence, tools such as StormGen can be used to assist the preliminary development of quick-and-dirty Storm topologies, e.g., to be further processed and improved with the aid of OSTIA.

In addition, several deployment modelling technologies may be related to OSTIA since their role is to model the deployment structure represented by Big data architectures so that it can actually be deployed using compliant orchestrators. One such example is Celar[6], a deployment modelling technology based on the TOSCA OASIS Standard[7]. Celar and related technologies (e.g., Alien4Cloud[8]) may be used in combination with OSTIA since their role is that of representing the infrastructure needed by modelled (Big data) applications so that they can be deployed. This representation is realised by means of infrastructure blueprints to be run by compliant orchestrators. The role of OSTIA in this scenario, is that of helping the quality refinement of an application topology to represent the very infrastructure needed for its run-time environment.

VII. Conclusion

Acknowledgment

The Authors' work is partially supported by the European Commission grant no. 644869 (EU H2020), DICE. Also, Damian's work is partially supported by the European Commission grant no. 610531 (FP7 ICT Call 10), SeaClouds.

References

[1] C. K. Emani, N. Cullot, and C. Nicolle, "Understandable big data: A survey." *Computer Science Review*, vol. 17, pp. 70–81, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/csr/csr17.html#EmaniCN15

[2] B. Ratner, *Statistical and Machine-Learning Data Mining: Techniques for Better Predictive Modeling and Analysis of Big Data*. CRC PressINC, 2012. [Online]. Available: http://books.google.de/books?id=0KuIJTAznLUC

[3] I. Ivkovic and M. W. Godfrey, "Enhancing domain-specific software architecture recovery." in *IWPC*. IEEE Computer Society, 2003, pp. 266–273. [Online]. Available: http://dblp.uni-trier.de/db/conf/iwpc/iwpc2003.html#IvkovicG03

[4] R. Evans, "Apache storm, a hands on tutorial." in *IC2E*. IEEE, 2015, p. 2. [Online]. Available: http://dblp.uni-trier.de/db/conf/ic2e/ic2e2015.html#Evans15

[5] M. Hirzel, H. Andrade, B. Gedik, G. Jacques-Silva, R. Khandekar, V. Kumar, M. P. Mendell, H. Nasgaard, S. Schneider, R. Soul, and K.-L. Wu, "Ibm streams processing language: Analyzing big data in motion." *IBM Journal of Research and Development*, vol. 57, no. 3/4, p. 7, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/ibmrd/ibmrd57.html#HirzelAGJKKMNSSW13

[6] D. Wang and J. Liu, "Optimizing big data processing performance in the public cloud: opportunities and approaches." *IEEE Network*, vol. 29, no. 5, pp. 31–35, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/network/network29.html#WangL15

[7] Y. Tamura and S. Yamada, "Reliability analysis based on a jump diffusion model with two wiener processes for cloud computing with big data." *Entropy*, vol. 17, no. 7, pp. 4533–4546, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/entropy/entropy17.html#TamuraY15

[8] K. Chandrasekaran, S. Santurkar, and A. Arora, "Stormgen - a domain specific language to create ad-hoc storm topologies." in *FedCSIS*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., 2014, pp. 1621–1628. [Online]. Available: http://dblp.uni-trier.de/db/conf/fedcsis/fedcsis2014.html#ChandrasekaranSA14

[6] https://github.com/CELAR/c-Eclipse

[7] https://www.oasis-open.org/apps/org/workgroup/tosca/

[8] http://alien4cloud.github.io/