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; (c) tackle the occurrence of common anti-patterns across considered topologies. 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 involved in designing and developing applications for Big data, e.g., Apache Storm[2], Apache Spark[3] or Hadoop[4] to name a few.

In our own experience with designing and developing for Big Data, we observed that a key complexity lies in evaluating the effectiveness of architectures, or better, *topologies* - in the Big data jargon. Effectiveness in Big data terms means being able to deploy architectures/topologies consistently with restrictions from necessary Big data frameworks involved (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 and Spark frameworks. We argue that such effectiveness can be maintained at design time thus easing deployability of Big data topologies and 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 step, OSTIA analyses the operating topology to make sure it is consistent with restrictions and best practices for the necessary Big data frameworks at hand. Also, in an effort to tackle said complexities and offer support in a in a DevOps fashion, OSTIA was engineered to act as a mechanism that closes the feedback loop between operating Streaming topologies (-Ops phase) and their development and improvement phase (Dev-phase).

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) checking that said topologies do not show any anti-patterns [5] that may lower performance and limit executability; (d) finally, export said topologies for further analysis, e.g., through verification techniques [6].

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

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

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 benefits we perceived in using it. Section VI elaborates further on the benefits by providing an actual evaluation of OSTIA using three cases from industrial practice. Section VII-B discusses the results and evaluation, also outlining OSTIA limitations and potential threats to its validity. Finally, Sections VIII and IX 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 structural quality of Big-Data streaming designs?"

The results contained in this paper in response to this research question were initially elaborated within a free-form focus group [7] involving three experienced practitioners and researchers on Big data streaming technology such as Storm. Following the focus group, through self-ethnography [8] and brainstorming we designed OSTIA and identified the series of consistency checks, algorithmic evaluations as well as anti-patterns that can now be applied through OSTIA while recovering an architectural representation for Storm topologies.

In addition, the OSTIA prototype we developed[5], was refined incrementally and evaluated through a series of case-studies [9] involving an industrial partner. The partner in question uses open-source social-sensing software do elaborate a Big-Data application that: (a) aggregates news assets from various sources (e.g., Twitter, Facebook, etc.) based on user-desired specifications (e.g., topic, sentiment, etc.); (b) presents and allows the manipulation of data. Said application is based on the SocialSensor App[6] which features the combined action of three complex streaming topologies using Storm (see Fig. 1 for a sample of said topologies).

More in particular, the topology in Fig. 1 extracts data from sources and manipulates said data to divide and arrange contents based on type (e.g., article vs. media), later updating a series of databases (e.g., Redis) with these elaborations.

Finally, we applied well-established verification approaches to integrate the value and benefits behind using OSTIA. More in particular, we engineered OSTIA to support exporting of elicited topologies for their further analysis using the Zot LTL model-checker [10] using the approach outlined in our previous work [6].

[5]https://github.com/maelstromdat/OSTIA
[6]https://github.com/socialsensor

III. Research Solution

@Pooyan,Andrea: here we should probably elaborate on OSTIA's architecture and the design principles that led us to define it as such... also we might want to elaborate on its components, the structure I'm suggesting below is merely tentative but it will give us ahead start!!

- OSTIA Architecture
- we should probably elaborate the architecture part (or on a separate "implementation" part or paragraph) with a link to the downloadable technology - @Andrea: can we bundle it up as plugin for Eclipse? E.g., somehow using RCP?
- OSTIA Antipatterns Module
- OSTIA Visualisation Module
- OSTIA extensibility
- OSTIA explanation of use and simple usage scenario

IV. Stream Design Anti-Patterns

This elaborates on the anti-patterns we elicited through self-ethnography. These anti-patterns are elaborated further within OSTIA to allow for their detection during streaming topology inference analysis.

A. Multi-Anchoring

In order to guarantee fault-tolerant stream processing, tuples processed by bolts needs to be anchored with the unique id of the bolt and be passed to multiple ackers int he topology. Therefore, ackers can keep track of tuples in the topology.

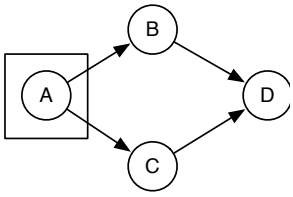

Fig. 2. Multi-anchoring.

B. Cycle in Topology

Technically it is possible to have cycle in Storm topologies. An infinite cycle of processing would create an infinite tuple tree and make it impossible for Storm to ever acknowledge that spout tuple. Therefore, tuple trees should terminate at some point.

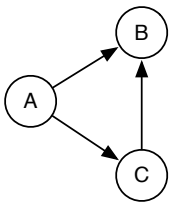

Fig. 3. Cycle n Topology.

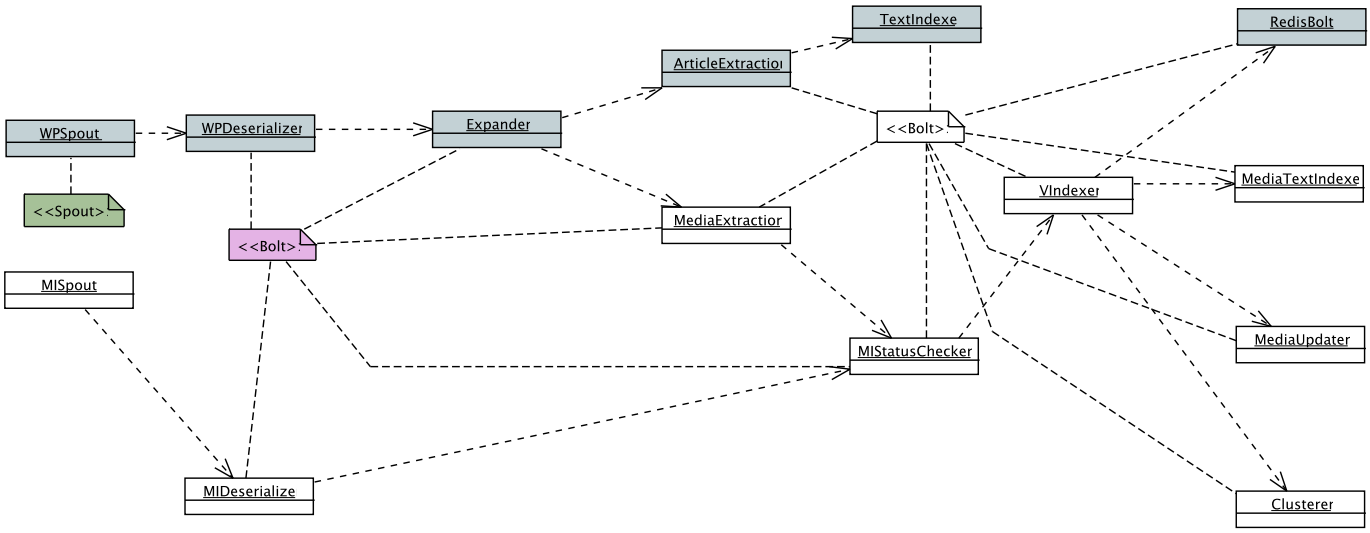

Fig. 1. A sample Storm topology in the SocialSensor App.

C. Persistent Data

If two processing elements wants to update a same entity in a storage, there should be a consistency mechanism in place.

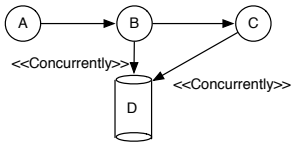

Fig. 4. Concurrency management.

V. Algorithmic Analysis on Stream Topologies

A. fan-in/fan-out

For Bolts, fan-in is the number of processing elements (Bolts and Spouts) that are connected to this specific bolt in the topology and fan-out is the number of PEs that are dependent to this specific bolt. For Spouts, fan-in is the number of topics (from Kafka) that this specific spout is subscribed to and fan out is the number of bolts that is dependent to this spout.

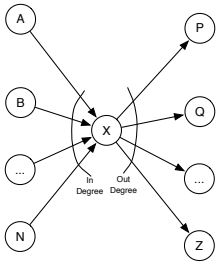

Fig. 5. Fan-in fan-out in Stream topologies.

B. topology cascading

By topology cascading, we mean connecting two different Storm topologies via a messaging framework like Kafka.

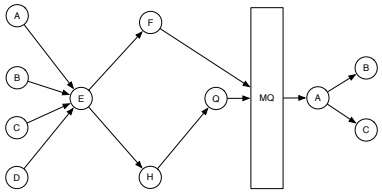

Fig. 6. cascading.

C. Topology clustering

Identifying the coupled processing elements and put the in a cluster in a way that elements in a cluster have high cohesion and less coupled with the elements in other clusters.

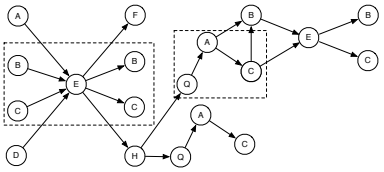

Fig. 7. clustering.

D. Computation funnel

When there is not a path from data source (spout) to the bolts that sends out the tuples off the topology to another topology through a messaging framework or through a storage.

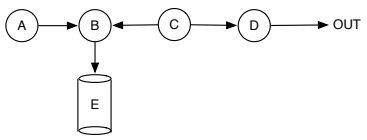

Fig. 8. funnel.

E. Linearizing a topology

Sorting the processing elements in a topology in a way that topology looks more linear visually.

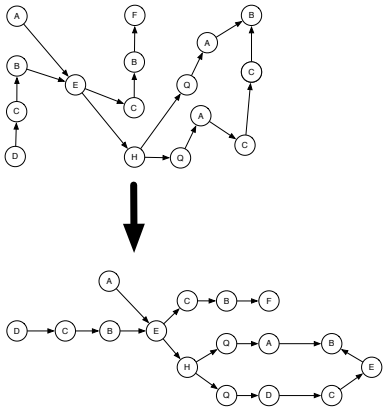

Fig. 9. linearizing.

VI. Evaluation

@Marcello,Francesco: we should also probably elaborate on the kind of verification technique we are using and how that can help in evaluating the topology.. remember here we do not have the DICE restriction so we can mention any kind of analysis that it would be possible to run, also analyses that are currently in the hands of other DICE partners!!

- we can use the ATC case study as much as we want - that yields already three topologies that we can infer
- ATC has agreed that we can mention their role in this exercise, I also showed them the topology that we elicited basically with OSTIA and they already made considerations on how to improve it
- in the evaluation we should also comment on how OSTIA can help you in visualizing the application topology that you may be considering to use by reusing a big-data application for something else... visualising the application topology and analysing it may allow you to improve it while you are using it as a starting point for your application
- another application that we can use is the one that NETF is considering for their own scenario, KILLRWEATHER - https://github.com/killrweather/killrweather
- 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

VII. 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

VIII. Previous and Related Work

The work behind OSTIA stems from the EU H2020 Project called DICE[7] 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 [11] 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

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

a performance [12] or reliability point of view [13]. 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.

We proposed BO4CO [14], an approach for locating optimal configurations using ideas of carefully choosing where to sample by sequentially reducing uncertainty in the response surface approximation in order to reduce the number of performance measurements. We have carried out extensive experiments with three different stream topologies running on Apache Storm. Experimental results demonstrate that BO4CO outperforms the baselines in terms of distance to the optimum performance with at least an order of magnitude.

Second, from a modelling perspective, approaches such as StormGen [15] 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 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.

Third, from a verification perspective, ...
@Marcello,Francesco: here we should probably elaborate on what kind of verification approach we are using and what other verifications may be done, e.g., using some related work at this point... e.g., is there any other verification attempt considering JSON as an interchange format? I would discuss these and compare them to OSTIA as a whole

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[8], a deployment modelling technology based on the TOSCA OASIS Standard[9]. Celar and related technologies (e.g., Alien4Cloud[10]) 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.

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

IX. Conclusion

Big data applications are rapidly gaining interest and momentum by small to big players on the market, even beyond the IT corner. Applications that may heavy use of Big data application frameworks require intensive reasoning on design aspects typically around the topology of basic operations to be applied while manipulating the data. In this paper we presented OSTIA, a tool to assist designers and developers in this reasoning campaign. OSTIA helps designers and developers improving the quality of their big data topologies by recovering and analysing the architectural topology on-the-fly. The analyses that OSTIA is able to apply focus on checking for the compliance to essential consistency rules specific to targeted big data frameworks. Also, OSTIA allows to check wether the recovered topologies contain occurrences of key anti-patterns we elicited and studied in our own experience and previous work. In addition, OSTIA allows to export the recovered topology using a common JSON format to encourage further verification using known verification and validation technology. By running a case-study with a partner organization, we observed that OSTIA assists designers and developers in establishing and improving the quality of topologies behind big data applications. In the future we plan to further our understanding of the anti-patterns that may emerge across big data topologies, e.g., learning said anti-patterns by using graphs analysis techniques inherited from social-networks analysis. Also, we plan to expand OSTIA to support further technologies beyond the most common application framework for streaming, i.e., Storm. Finally, we plan to further evaluate OSTIA using more ad-hoc empirical evaluation campaigns in industry.

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

[9]https://www.oasis-open.org/apps/org/workgroup/tosca/
[10]http://alien4cloud.github.io/

[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] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, *Pattern-oriented software architecture. Vol. 2, Patterns for concurrent and networked objects*. John Wiley, Chichester, England, 2000.

[6] M. M. Bersani, S. Distefano, L. Ferrucci, and M. Mazzara, "A timed semantics of workflows." in *ICSOFT (Selected Papers)*, ser. Communications in Computer and Information Science, A. Holzinger, J. Cardoso, J. Cordeiro, T. Libourel, L. A. Maciaszek, and M. van Sinderen, Eds., vol. 555. Springer, 2014, pp. 365–383. [Online]. Available: http://dblp.uni-trier.de/db/conf/icsoft/icsoft2014s.html#BersaniDFM14

[7] D. L. Morgan, *Focus groups as qualitative research*. Sage Publications, Thousand Oaks, 1997.

[8] M. Hammersley and P. Atkinson, *Ethnography*. London: Routledge, 2003.

[9] S. B. Merriam, *Case study research in education: A qualitative approach*. Jossey-Bass Publishers, 1991, vol. 1, no. 1980. [Online]. Available: http://psycnet.apa.org/psycinfo/1989-97006-000

[10] C. A. Furia, D. Mandrioli, A. Morzenti, and M. Rossi, "Modeling time in computing: A taxonomy and a comparative survey," *ACM Comput. Surv.*, vol. 42, no. 2, pp. 6:1–6:59, Mar. 2010. [Online]. Available: http://doi.acm.org/10.1145/1667062.1667063

[11] 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

[12] 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

[13] 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

[14] P. Jamshidi and G. Casale, "An uncertainty-aware approach to optimal configuration of stream processing systems." in *VLDB (under evaluation)*, 2016.

[15] 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