Experiences and Challenges in Building a Data Intensive System for Data Migration*

Marco Scavuzzo
Elisabetta Di Nitto
Danilo Ardagna
Politecnico di Milano, Dipartimento di Elettronica Informazione e Bioingegneria
Milano, Italy
name.lastname@polimi.it

Recent analyses[2, 4, 5] report that many sectors of our economy and society are more and more guided by data-driven decision processes (e.g., health care, public administrations, etc.). As such, *Data Intensive (DI) applications* are becoming more and more important and critical. They must be fault-tolerant, they should scale with the amount of data, and be able to elastically leverage additional resources as and when these last ones are provided [3]. Moreover, they should be able to avoid data drops introduced in case of sudden overloads and should offer some Quality of Service (QoS) guarantees.

Ensuring all these properties is, per se, a challenge, but it becomes even more difficult for DI applications, given the large amount of data to be managed and the significant level of parallelism required for its components. Even if today some technological frameworks are available for the development of such applications (for instance, think of Spark, Storm, Flink), we still lack solid software engineering approaches to support their development and, in particular, to ensure that they offer the required properties in terms of availability, throughput, data loss, etc. In fact, at the time of writing, identifying the right solution can require several rounds of experiments and the adoption of many different technologies. This implies the need for highly skilled persons and the execution of experiments with large data sets and a large number of resources, and, consequently, a significant amount of time and budget.

To experiment with currently available approaches, we performed an action research experiment focusing on developing-testing-reengineering a specific DI application, Hegira4Cloud (H4C), that migrates data between widely used NoSQL databases, including so-called *Database as a Service* (DaaS), as well as on-premise databases. This is a representative DI system because it has to handle large volumes of data with different structures and has to guarantee that some important characteristics, in terms of data types and transactional properties, are preserved. Also, it poses stringent requirements in terms of correctness, high performance, fault tolerance, and fast and effective recovery.

In our action research, we discovered that the literature offered some high level design guidelines for DI applications and some tools to support modelling and QoS analysis/simulation of complex architectures. However, the available tools were not yet suitable to support DI systems. Moreover, we realized that the available big data frameworks we could have used were not flexible enough to cope with all possible application-specific aspects of our system.

Hence, to achieve the desired level of performance, fault tolerance and recovery, we had to adopt a time-consuming, experiment-based approach [1, 6], which, in our case, consisted of three iterations: (1) the design and implementation of a Mediation Data Model capable of managing data extracted from different databases, together with a first monholitical prototype of H4C; (2) the improvement of performance of our prototype when managing and transferring huge amounts of data; (3) the introduction of fault-tolerant data extraction and management mechanisms, which are independent from the targeted databases.

Among the others, an important issue that has forced us to reiterate in the development of H4C concerned the DaaS we interfaced with. In particular these DaaS, which are well-known services with a large number of users: (1) were missing detailed information regarding the behaviour of their APIs; (2) did not offer a predictable service; (3) were suffering of random downtimes not correlated with the datasets we were experimenting with.

In this journal first presentation, we describe our experience and the issues we encountered that led to some important decisions during the software design and engineering process. Also, we analyse the state of the art of software design and verification tools and approaches in the light of our experience, and identify weaknesses, alternative design approaches and open challenges that could generate new research in these areas. More details can be found in the journal publication.

REFERENCES

[1] R. Baskerville and M.D. Myers. 2004. Special Issue on Action Research in Information Systems: Making is Research Relevant to Practice Foreword. *MIS Quarterly* 28, 3 (2004), 329–335.
[2] P. Chen and C. Zhang. 2014. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences* 275 (2014).
[3] K. Kambatla et al. 2014. Trends in big data analytics. *J. Parallel and Distrib. Comput.* 74, 7 (2014), 2561 – 2573. Special Issue on Perspectives on Parallel and Distributed Processing.
[4] J. Manyika et al. 2012. Big data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Global Institute. (2012).
[5] B. Marr. 2015. Big Data: 20 Mind-Boggling Facts Everyone Must Read. http://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read. (2015).
[6] R. O'Brien. 1998. An overview of the methodological approach of action research. *Theory and Practice of Action Research* (1998).

*Journal first presentation of the paper: Scavuzzo, M., Di Nitto, E. & Ardagna, D. Empir Software Eng (2017). https://doi.org/10.1007/s10664-017-9503-7

ICSE '18, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5638-1/18/05.
https://doi.org/10.1145/3180155.3182534