Towards Reproducible Evaluation of Large-Scale Distributed Systems

Miguel Matos
INESC-ID & IST. U. Lisboa
Lisbon, Portugal
miguel.marques.matos@tecnico.ulisboa.pt

ABSTRACT

Reproducing experimental results is nowadays seen as one of the greatest impairments for the progress of science in general and distributed systems in particular. This stems from the increasing complexity of the systems under study and the inherent complexity of capturing and controlling all variables that can potentially affect experimental results. We argue that this can only be addressed with a systematic approach to all the stages of the evaluation process. This raises the following challenges: i) precisely describe the environment and variables affecting the experiment, ii) minimize the number of (uncontrollable) variables affecting the experiment and iii) have the ability to subject the system under evaluation to controlled fault patterns. In the following, we highlight the research directions we are currently pursuing to address these goals. Our overarching goal is to build an open-source evaluation platform, Angainor [1], able to deploy an experiment, control the network topology, inject faults, monitor the whole experiment and automatically derive summary statistics of the experimental data.

Keywords

distributed systems, systems evaluation, reproducibility

1. RESEARCH STATEMENT

> Science moves forward by corroboration - when researchers verify others' results. Science advances faster when people waste less time pursuing false leads. No research paper can ever be considered to be the final word, but there are too many that do not stand up to further study.
>
> *Challenges in irreproducible research*
> *Nature, June 2016*

Experiment reproducibility is thus crucial for science allowing one, as Isaac Newton purportedly stated, to stand on the shoulders of giants.

This is true for science in general, but particularly relevant for distributed systems. In fact, as our society becomes increasingly digital, the need to have robust and trustworthy distributed systems is increasingly important. As a matter of fact, at the heart of many modern enterprises such as Cloud Computing, Big Data Systems, Internet of Things and Blockchain lies one or more distributed system. However, as systems become more complex, recreating experimental conditions with some precision becomes increasingly difficult. This stems from the increasing complexity of the systems themselves which are made of several distinct components, which in turn have specific configurations, software versions and dependencies on other components, libraries,

BIO.

Miguel Matos is Assistant Professor at Instituto Superior Técnico, Universidade de Lisboa, Portugal and Senior Researcher at INESC-ID, in the Distributed Systems Group. His research interests lie in the area of distributed systems, in the subjects of scalability, consistency, scalability, and performance. In particular, he is conducting research in blockchain and related problems, consistency in large-scale databases, and automation and reproducibility of performance and scalability evaluations in large-scale systems. He has several publications in top journals and conferences, such as TPDS, JPDC, Eurosys, Middleware, IPDPS, and SRDS. He is currently the PI of an ongoing research project on reproducible evaluation of large-scale distributed systems funded by the Portuguese National Science Foundation (FCT). In the past, he co-founded MIMA Housing and was Senior Engineer at LeanXcale, a startup developing large-scale database solutions.

[1]For further details check http://angainor.science

ApPLIED '18 July 27, 2018, Egham, United Kingdom

ACM ISBN 978-1-4503-5775-3/18/07.

DOI: https://doi.org/10.1145/3231104.3231113

and environment variables. The presence of the network with a concrete topology, fault model, and configuration further aggravates the problem. Besides, evaluating a distributed system under different fault patterns at the process and network level is key to understanding their behavior and resilience under realistic and adversarial conditions. Expressing all of these conditions about the system and its environment in a concise and precise manner is extremely difficult.

1.1 Approach

Addressing these issues implies addressing the technical aspects that hamper reproducibility: i) experiment complexity and variability; ii) the precise exposition of the system under evaluation; and iii) the use of appropriate methods and tools to reason about the experimental results.

Each one of these aspects has been identified recently as leading causes for irreproducibility in empirical evaluations in software systems [3]. Any modern system is a complex construction composed of several software packages - from the operating system and libraries to the application itself - and their respective configurations. In distributed systems, this is further aggravated by the network, which poses another axis of variability, but also by the possibility of faults in different parts of the system which need to be considered in a comprehensive evaluation. Capturing all these variables in a research paper is not only extremely hard but also quite impractical due to the usual space constraints. In fact, we can draw a parallel between what is reported in a paper and the actual experiment, with the mismatch that exists between code and its documentation. The latter is often outdated, imprecise and lacking in detail to help the reader reason how the system is supposed to behave. We argue that this can only be solved by providing, along with the research paper exposing the core ideas, a precise description of the software artifacts, configurations, environment, and benchmarks. Naturally, the characteristics of the environment where the evaluation takes place unquestionably affect the behavior and performance of the system, for instance, the underlying hardware and operating system version. Other characteristics such as concrete configurations often have a serious impact on performance but expressing them in detail is hard without actually providing the actual configuration file. In general, these are known variables that the researcher is typically aware of and, with due diligence, can be clearly articulated when exposing the experiment. However, there are many other variables, that can affect system behavior and performance of which one is not aware of - the known unknowns - that can have a profound impact on the results. For instance, it has been shown that the size of the environment variables significantly affects the performance of several low-level C benchmarks [5]. While apparently unrelated, this variance stems from the memory alignment that the compiler is able to do in certain environments.

Therefore, two independent researchers reproducing this experiment without considering this seemingly unimportant variable can come to quite different conclusions. When we extrapolate this to a distributed system, with several processes, the network and the existence of a given fault pattern, it is actually surprising that we can reproduce results at all. For instance, a simple fault injection that corrupts a single data block on disk can lead to serious consequences in several databases systems, including unavailability, data corruption and data loss [4]. Only by systematically pruning these sources of variability - both known and unknown - can we hope to improve experiment reproducibility.

Our current research is a step in this direction. Before proceeding, we need to consider two important points. First, variability cannot ever be fully eliminated in a real system as, for instance, the decisions done by the scheduler in an operating system or network router are often unpredictable. Some works try to overcome this by building custom operating systems that enforce determinism [2] with specific APIs and programming models. However, the API is not generic and imposes several limitations on distributed programs, preventing its general usage. In fact, a fully deterministic execution can only be achieved in a simulated environment. Unfortunately, simulators have to make simplified assumptions about reality, often requiring applications to be written specifically for the simulator. As the implementation is different, and the environment is significantly simplified, this results in drawing conclusions in the simulated environment that might not be aligned with the real observations. Second, variability is important to evaluate a system under different conditions but only to the extent that such variability is well-known and controlled. If several parameters about the system and environment vary simultaneously, then it is impossible to tell with confidence the real impact of what is being intentionally varied [3]. Finally, two additional aspects need to be considered in the quest for reproducible experiments: the measurement tools (benchmarks) and the measurements themselves. Different tools often provide different measurements about the same aspect of the system thus potentially leading two researchers to two different conclusions. For instance, several Java profilers are known to provide very different measurements about the same aspect of a Java program [6]. This means that the same due diligence that needs to be done when describing the system being evaluated also needs to be applied to the set of tools used for the evaluation. Additionally, the raw measurements collected during the experiment rarely can be understood without some consolidation through statistical analysis. However, simple statistics such as mean and standard deviation often induce in error as its adequacy depends heavily on the raw data distribution. This is well illustrated by the Anscombe's quartet - a set of four data sets with the same statistical properties such as mean, variance and Pearson correlation but with strikingly different distributions [1]. Once again, this can lead researchers to draw incorrect conclusions and claims, unless the proper summary statistics and plots are used.

Currently, we are taking a systematic and principled approach to distributed systems experimentation and evaluation. The major goal is to reduce experiment variability in the process, network and fault pattern levels. To this end, we are building a platform, Angainor, for reproducible evaluation and fault injection in distributed systems and a domain specific language to precisely express the system under evaluation. At the process level, we will address this issue by relying on container technology. Containers are a lightweight self-contained mechanism to describe an operating system, environment variables, libraries, software packages and configurations of a single process, allowing to remove ambiguity, and hence variability, at the process level. This will be extended with a domain specific language to express dependencies among processes (containers). At the network level,

we will develop a domain specific language to precisely describe the network topology and its characteristics (for instance available bandwidth and round-trip time). The goal is to provide an extensible library of fault patterns that can be applied to processes (for instance a process crash) and the network (for instance packet drops) following synthetic or real patterns. Finally, we also want to systematize experiment monitoring and provide tools for semi-automated modeling and statistical analysis of the experimental data which is key to properly reason about the observed results.

By systematically covering all aspects of evaluation in a single platform - from experiment design to result analysis - we plan to greatly simplify designing proper reproducible experiments. The platform, called Angainor, will be made available as open source to promote adoption. We believe this approach will greatly simplify researchers' life - a key to adoption - but also allow researchers to verify others' results, allowing science to move forward by corroboration.

Acknowledgments.

This work was partially supported by Fundação para a Ciência e Tecnologia (FCT) through the project with reference LISBOA-01-0145-FEDER-031456.

2. REFERENCES

[1] F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1), 1973.

[2] A. Aviram, S.-C. Weng, S. Hu, and B. Ford. Efficient system-enforced deterministic parallelism. *Commun. ACM*, 55(5), 2012.

[3] S. Blackburn et al. The truth, the whole truth, and nothing but the truth: A pragmatic guide to assessing empirical evaluations. *ACM Trans. Program. Lang. Syst.*, 38(4), 2016.

[4] A. Ganesan, R. Alagappan, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Redundancy does not imply fault tolerance: Analysis of distributed storage reactions to single errors and corruptions. In *Proceedings of the 15th Usenix Conference on File and Storage Technologies*, FAST'17, 2017.

[5] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney. Producing wrong data without doing anything obviously wrong! *SIGPLAN Not.*, 44(3), 2009.

[6] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney. Evaluating the accuracy of java profilers. In *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '10, 2010.