Experiment - Ownership using Persmissioned Blockchains

Robert Diebels

As part of a Master graduate project at the Universiteit van Amsterdam

May, 2017

Abstract

This article describes an experimental design which main focus is gathering and comparing data from the implementation of a Performing Rights Organizations (PROs) use-case in several permissioned blockchains systems. Once implemented the outcome of an analysis described in this article will be used to evaluated a master graduate-projects' hypothesis on permissioned blockchains performance compared to current PRO systems.

Index terms— thesis, permissioned blockchains, experiment, design, setup

1 Introduction

As shown in [4] current literature on permissioned blockchains primarily focuses on defining problem areas and providing solutions. None of the articles on permissioned blockchains contained a controlled experiment aimed at comparison of current permissioned blockchain implementations. One of the articles found in [4] concludes with the statement that it was: "hard to find empirical evidence to show the comparison between blockchain approaches and traditional approaches." [6].

The graduate-project of which the experiment defined in this article is part of aims to fill that void. This is achieved by implementing a use-case defined by several Performing Rights Organizations (PROs) from the music-industry in a multitude of current permissioned blockchains. The implementations are run in a controlled environment, data is gathered on the performance of an implementation and subsequently compared to other implementations and data from PROs systems.

In the past surveys [5, p.746] have identified a lack of replication of Software Engineering (SE) experiments. The design of this experiment as described in section 2 has been structured in such a way that replication should be easily achieved. Details of the PROs use-case are described in section 3. Finally section 4 describes the data's life-cycle of collection, processing and analysing the data gathered from the experiment.

2 Experimental design

This experiment evaluates the validity of the graduate projects' hypothesis. This section specifies the graduate projects experimental design used for evaluation and which studies used in determining the design.

Hypothesis Permissioned blockchains outperform current ownership-validation systems in terms of throughput and verification-speed.

2.1 Experimentation

Evaluating hypothesis validity is achieved by measuring aspects of blockchain implementations determining their performance. The experiment defined in [1, p.64] is used as the basis for the graduate projects' experimental design.

As in [1] the experiment encompasses the evaluation of (1) an implementation of the use case without faults, (2) a fault-tolerance implementation where every 3 seconds a randomly selected validating node is crashed, (3) and a network-latency tolerance implementation, where time-outs are introduced before every read and write.

2.2 Variables

Independent The following independent variables are part of the experiment. A number of variables depend on external factors. I_{min_val} is dependent on the minimum number of validating nodes demanded by the protocol of a blockchain implementation and transaction-size is dependent on the Minimum Viable Data (MVD) of the use-case.

1. CPU speed (GHz)
2. RAM (GiB)
3. Number of validating nodes (N_{val}) (I_{min_val}, 64)
4. Number of faulty nodes (N_{fault})
5. Block-size (MB) (128, 32768)
6. Transaction-size (KB) (MVD)
7. Run-time (200 blocks)

Dependent The following dependent variables are to be measured for performance evaluation.

1. Throughput, transactions per second (Tx/s).
2. Latency, block-formation per second (Bf/s).

Possible confounding factors Network latency? . . .

2.3 Implementations

Inclusion In [3] recommendations are made for software engineering researchers to ensure their algorithms are reproducible. Those recommendations and the suggestion made in [2] to use docker containers serve as the base criteria for an implementations inclusion into the experiment.

The recommendations used as inclusion criteria are I, II, IV, V, VII and X. A description of all recommendations from [3] can be found in Table 2.

The implementations found in [4] will be evaluated and included in the experiment if found they fulfill the inclusion criteria.

2.4 Deployment

Hardware Each node in the network will be hosted on a separate Amazon EC2 t2.medium instance. The t2.medium instance hardware specifications related to independent variables can be found in Table 1.

Independent variable	Instance value
Computing instance	t2.medium
CPU Speed	(0, 3.3) Ghz
RAM	4 GiB
Storage	Amazon EBS gp2
Volume Size	1 GiB - 16 TiB
Throuput Max	160 MiB/s[1]

Table 1: Amazon EC2 hardware specifications

Containerization All blockchain implementations are containerized using docker and their images are placed in the Docker Hub. The images will be made available at `https://hub.docker.com/u/robertdiebels/`.

[1] t2.medium has lower throuput then defined on the Amazon EBS page. How much is unclear.

3 Use-case

Need to contact SENA and BUMA for this section.

3.1 Entities

Describe which types of entities take part in the blockchain.

Artists(rights-holders/owners), distributors(users), ...

3.2 MVD

Describe the minimum amount of data/information for a distribution of rights/ownership needed.

Rights-holders, song, album, monetary distribution among rights-holders, ...

3.3 Transactions

Describe types of transactions done by SENA and BUMA, the amount of transactions per year etc..

3.4 Hardware

Provide an overview of hardware attained from SENA and BUMA here (if they agree to publish that).

4 Analysis

This section explains where, how and what data will be collected and how that data will be processed and analyzed.

4.1 Data collection

Location Data related to both independent and dependent variables will be gathered from each node participating in the block-chain network.

Procedure Detailed on a per implementation basis. Procedure details will be reported after the use-case has been implemented in each permissioned blockchain.

4.2 Data processing

Output Gathered data-dumps are processed on each node and transformed to a JSON output format. A full verion of the output format can be found in Appendix B.

4.3 Approach

Samples Samples are collected from 3 or more permissioned blockchains.

Pairing Samples are unpaired as the samples are taken from different permissioned blockchain implementations which are unrelated an of which sample sizes may vary.

Gaussian To determine if the samples have a normal-distribution (Gaussian) the D'Agostino-Pearson normality test will be used.

Test Depending on the results of the normality test either a Kruskal-Wallis or a one-way ANOVA test will be applied to determine the difference in means of one independent variable. Whe measuring two independent variables either a two-way ANOVA or a Scheirer-Ray-Hare test will be applied.

References

[1] Ethan Buchman. *Tendermint: Byzantine Fault Tolerance in the Age of Blockchains*. PhD thesis, 2016.

[2] Jürgen Cito, Vincenzo Ferme, and Harald C Gall. Using docker containers to improve reproducibility in software and web engineering research. In *International Conference on Web Engineering*, pages 609–612. Springer, 2016.

[3] Tom Crick, Benjamin A Hall, and Samin Ishtiaq. " can i implement your algorithm?": A model for reproducible research software. *arXiv preprint arXiv:1407.5981*, 2014.

[4] Robert Diebels. Literature survey - ownership using persmissioned blockchains. apr 2017.

[5] Dag IK Sjøberg, Jo Erskine Hannay, Ove Hansen, Vigdis By Kampenes, Amela Karahasanovic, N-K Liborg, and Anette C Rekdal. A survey of controlled experiments in software engineering. *IEEE transactions on software engineering*, 31(9):733–753, 2005.

[6] Huaiqing Wang, Kun Chen, and Dongming Xu. A maturity model for blockchain adoption. *Financial Innovation*, 2(1):12, 2016.

Appendices

A Recommendations

Recommendation	Description	Used as criteria
I	"[a paper] must describe the algorithm in such a way that it is implementable by any reader of that algorithm.". This recommendation is interpreted as the state of the documentation of an implementation.	Yes
II	"We recommend that code be published under an appropriate open source license...". This recommendation is interpreted as an implementation being open-sourced and open to modification for personal use.	Yes
III	"[we] recommend that basic programming and computational skills are taught as core at undergraduate and postgraduate level.". This recommendation was not used as an inclusion criteria. The aim the experiment is to compare implementations. Educational backgrounds are not considered.	No
IV	"The use of a principled, high-level programming language in which to write your software helps hugely with the maintainability, robustness and openness of the software produced.". Interpreted as an implementation being written in a commonly used high-level language in the industry. [Java, Go, C#, C++]	Yes
V	"Testing new complex scientific software is difficult – until the software is complete, unit tests may not be available. You should thus aim to link to/from publicly-shared code: shared code is inherently more test-able.". Interpreted as an implementations test infrastructure being clearly defined.	Yes

VI	"Code should always include links to papers publishing key algorithms and the code should include explicit relationships to other projects on the repository (i.e. Project B was branched from Project A).". Not used. Evaluating proper referencing in implementation code is not within the experiments' scope.	No
VII	"Providing the source code of the tool helps, of course. But you must also provide details of precisely how you built and wrote the software.". Interpreted as the presence of docker containers and build-documentation.	Yes
VIII	"Avoid creating new representations when common formats already exist. Use existing extensible internationally standardised representations and formats to facilitate sharing and re-use." Not used. Evaluation of representations within code is not within the scope of this experiment.	No
IX	"Benchmarks should be public. They should allow anyone to contribute, implying that the tests are in a standard format." Benchmarks are optional as the experiment sets out the evaluate performance in a self-defined use-case.	Optionally
X	"The Web and the cloud really do open up a whole new way of working...". Interpreted as deploy-able to cloud infrastructure.	Yes

Table 2: Definitions of recommendations by [3] and their usage as inclusion criteria.

B Output format

```
{
    "runtime": {
                    "start": "<time-stamp in milliseconds>",
                    "stop": "<time-stamp in milliseconds>"
                },
    "blocks":  [
                    {
                        "hash":       "<Hash>",
                        "creation":   "<time-stamp in milliseconds>",
                        "transactions": [
                                            {
                                                "type": "<String>"
                                            }
                                        ]
                    }
                ]
}
```

Listing 1: JSON Output