

Analytical Comparison of DLT Platforms Activity

Oleg Iakushkin(✉), Anna Fatkina, Dmitry Selivanov, and Vladimir Korkhov

Saint-Petersburg State University, St. Petersburg, Russia
o.yakushkin@spbu.ru

Abstract. This paper describes a method for analyzing and comparing the activity of various distributed ledger platforms. The presented topics are the criteria for evaluating platforms, the method of their ranking and methods for automated collection information about them. A system that allows automatic assessment and ranking of distributed ledger technologies (DLT) with an open GitHub repository has been developed. Visualization of the collected information by means of Elasticsearch and Kibana is described.

Keywords: Analytics · GitHub · Distributed ledger

1 Introduction

There are a number of studies aimed at comparing technologies of distributed ledger and aggregation information about them [1,3,4]. Nevertheless, the distributed ledger field is dynamic, and the information that was relevant yesterday will not be true tomorrow. In this regard, we want to create a system that allows getting the most relevant, systematically updated information. The collection of information and its visualization are carried out automatically. Automation allows not only to provide up-to-date data but also to scale the system to any number of technologies under consideration.

Also, our aim is to present the collected information in a visual and applicable form. A system that makes it possible to automatically evaluate each technology according to several criteria has been developed. The according criteria were designed to provide versatile and objective information within the framework of data that can be obtained from open-access GitHub repositories.

Figure 1 shows the field of distributed ledger analysis. The financial assets area is covered by exchanges. To analyze private blockchains there is a framework called Blockbench [9]. Consensus protocols are considered and described by KPMG, see [2]. Smart contracts field is analyzed in [5,10].

The system described in this paper covers the area of community activity. For a more detailed study of technologies, we provide a visualized comparison of repositories for a number of parameters. On the basis of the chosen parameters, these technologies are estimated.

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11620, pp. 556–566, 2019.
https://doi.org/10.1007/978-3-030-24296-1_44

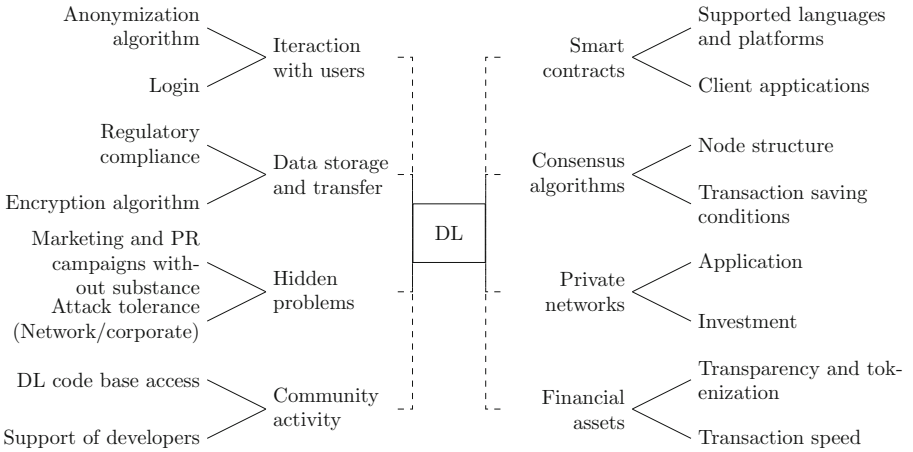

Fig. 1. Distributed ledger analysis. DL—distributed ledger.

2 Criteria and Rating Scale

To evaluate and compare distributed ledger platforms a system of criteria has been developed. According to these criteria, the activity of developing and supporting open source projects is evaluated, based on the corresponding GitHub repositories. The ranking is made on the following grounds:

1. *Developers Activity.* The assessment considers data of 3 months preceding the time of assessment.
   - The average number of code additions per week,
   - The average number of commits per week,
   - Last release date. An inverted scale is used for this item, that is, repositories with older releases are considered less active.
2. *Feedback activity* is shown by the rate of issue closure.
   For each issue i that has at least one comment feedback activity `FA` reads

$$\texttt{FA} = \frac{1}{n}\sum_{n}(\texttt{closed_at}(i) - \texttt{created_at}(i)),$$

   where n is a number of commented issues, `closed_at`(i) and `created_at`(i) are dates of closing and creating of issue i.
   In this criterion, we consider only those issues in which there is at least one comment. Such a condition is imposed in order to avoid counting those issues that developers put for themselves or for those that are closed beyond prescription and non-relevance.
3. *User interest* is made up of
   - Number of forks,

- The number of developing forks. By developing forks, we mean those that do not have the same date of creation and the date of the update. This item shows the interest of users in the development of the product or creating their own based on it.
- Number of issues created in the last 3 months (including open issues),
- The number of stars.

4. *Progress* consists of
   - The growth or decline ratio of the development activity is calculated by the formula

$$D_1 = \frac{\frac{1}{n_3}\sum_{j_3=1}^{n_3} \texttt{commits_per_week}(j_3)}{\frac{1}{n_1}\sum_{j_1=1}^{n_1} \texttt{commits_per_week}(j_1)},$$

     where n_1 and n_3 are numbers of weeks in one and three months, respectively, $\texttt{commits_per_week}(j)$ is a number of commits in a week j.
   - Growth or decline ratio of the development team

$$D_2 = \frac{\frac{1}{n_3}\sum_{j_3=1}^{n_3} \texttt{committers_per_week}(j_3)}{\frac{1}{n_1}\sum_{j_1=1}^{n_1} \texttt{committers_per_week}(j_1)},$$

     where n_1 and n_3 are numbers of weeks in one and three months respectively, $\texttt{committers_per_week}(j)$ is a number of contributors who made any commit in a week j.

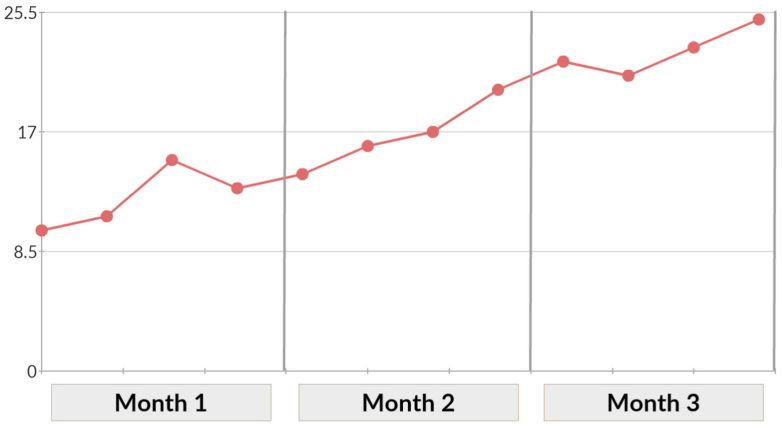

Fig. 2. An Example of growth ratio of activity or number of developers. Avarage for 3 months is 17.25, for the last month is 22.75. Growth ratio is 1.3.

Note that the first two subclauses of criterion 1 are similar. We take both of these points into account because we want to get a measure that does not depend on the work style of developers with GitHub. For example, the presence of a significant number of commits with smaller code changes in the repository

will not lead to a high estimate. Also, on the contrary, in the case of active work of developers who rarely make commits, the rating will not be underestimated. To obtain a high or low estimate of developer activity, both of these values have to be considered.

The fourth criterion shows the dynamics of the repository. It compares the data of the last three months with those of the last month. Figure 2 clearly demonstrates how their attitude reflects the growth or decline of the indicator. The value of the coefficient depends on the trend of the corresponding indicator.

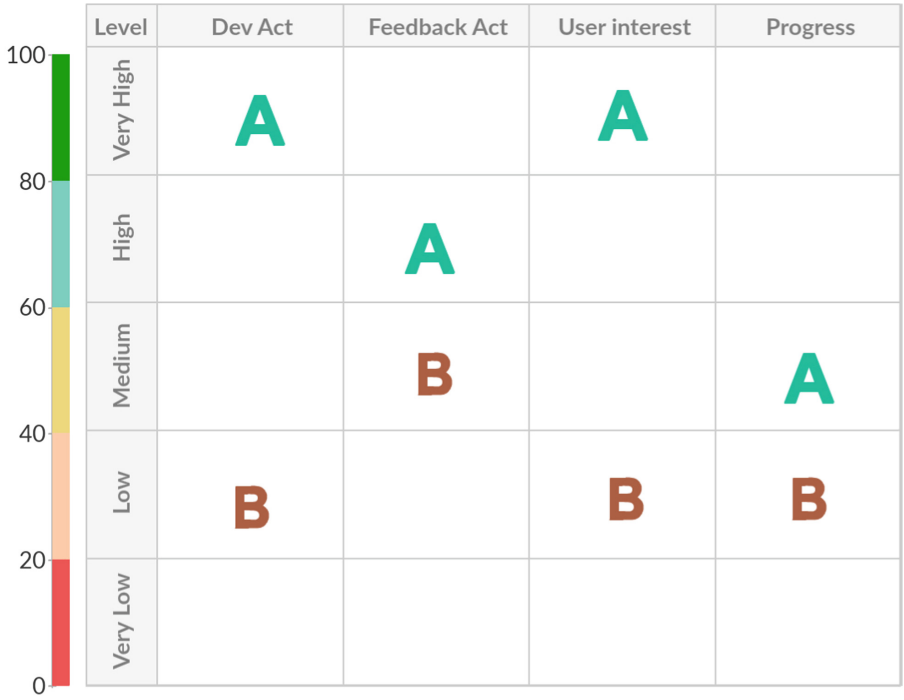

Fig. 3. Scale of ranking criteria values. A and B are projects to be evaluated.

The rating scale for each of the criteria is presented in Fig. 3. It is dynamic, depending on the values in the database. Thus, the ledger score shows the rank of a specific repository among those considered.

The scale has five levels, from *very low* to *very high*. To determine which level the value of a particular criterion belongs to, it is first scaled 100%, where 100% is the maximum value among all available data and 0% is the minimum. The 100% scale is divided into 5 intervals, each of which includes the upper limit, and the first one also includes 0%.

The ranking takes place automatically via an implemented scoring algorithm. For each considered GitHub repository, a database query is made and the values corresponding to each criterion are calculated. The value of the combined

criteria is considered equal to the average of the values of all sub-items of the corresponding criteria. After counting the values of all repositories, they are sorted in ascending order by each of the categories. The rating for the exact repository is set depending on the level to which the scale corresponds to.

3 Automated Information Collection System

We developed an automated information collection system. It obtains public data from repositories of distributed ledger platforms from GitHub using the Github REST API v3. The content and type of the collected data depend on whether the repository is the main project or a satellite project, e.g. for major projects, the following information is collected:

1. Repositories Name, owner, number of stars and subscribers, creation date, last modified date, wiki availability;
2. The branches of the project and their number. Some of the data obtained for each branch are: name, SHA of the last commit, URL of the last commit, whether the branch is protected (protected true/false);
3. Releases, including name, tag, author and date of each release, the total number of project releases;
4. Forked projects, their total. And also for each of the forks: name, owner, creation date, last commit date, number of stars/subscribers, etc .;
5. Closed and open pull requests, their number. Including for each of the pull requests: name, author, date of creation, date of last change, number of requested reviewers, current status;
6. Issues, their number. For each of the issues: name, author, creation date, last modified date, current status, a number of comments;
7. Commit activity. Represents data on the number of commits per day, for all days in the last year;
8. The code frequency of the project. It represents the data of the number of additions and deletions per week, for all weeks over the last year.
9. Contributors, their number, number of contributions for each contributor;
10. The frequency of changes in the project for the top 100 contributors. It represents the data of the number of additions and deletions per week for each of the contributors included in the top 100 by the number of commits for all weeks in the last year;
11. Commits, total in all branches, for a certain period of time (by default for the last 90 days). Including for each of the commits: sha, description, author, date of creation, branch of development, sha of parent commit, number of comments;
12. Commits in the master branch, for all the time. Individual information about each commit is the same as described above;
13. The status of CI builds for all commits for a certain period (default for the last 90 days). For each build: status (failed/successful/pending), title, author, creation date, last modified date.

For satellite projects, the following items are collected:

- Information about commit activity for the last year;
- The number of contributors, as well as the number of contributors for each of them;
- Information about the frequency of changes of the project (code frequency), i.e. the number of additions and deletions per week, for each week for the last year;
- The number of forks, as well as the number of stars/subscribers for each of the forks.

Because information collection about repositories of distributed ledger technologies from the GitHub platform is limited to the maximum allowable number of requests to the GitHub REST API v3 (by default, 60 requests per hour), a proxy server system is used. Therefore, receiving data about repositories occurs in multi-threaded mode. When one of the threads exhausts the allowable amount of requests, it starts using a new proxy server.

In addition, the proxy server is individual for each stream and, according to the GitHub API v3 documentation, performs no more than 60 requests per hour [6]. The software implementation of this uses *request* and *multithreading* libraries. The first of them allows you to create session objects that can use proxy server settings. URL requests are then made through these sessions. The second library is used to implement the multithreading approach.

4 Visualization

4.1 Kibana Visualize and Dashboard

For more convenient monitoring of the project activity, the collected data is arranged in charts. Some charts are presented in Fig. 4

Graphs are built using the Kibana [7] data visualization tool (Visualize & Dashboard). They show values for all downloaded data at the time of viewing. In other words, the data in the graphs are updated automatically.

For various repositories, graphs of compared values are presented, such as:

- The number of contributors and stars in the repository,
- Number of commits per year
- The number of open issues
- Number of forks and releases,
- The dynamics of the creation of pull requests (the total number and the number of those which were merged into the master branch)
- Dynamics of adding commits and closing issues,
- Dynamics of additions and deletions of lines of code.

The Kibana Dashboard provides interactive graphics. By clicking on the value of the repository or on its name in the legend of charts, the corresponding data

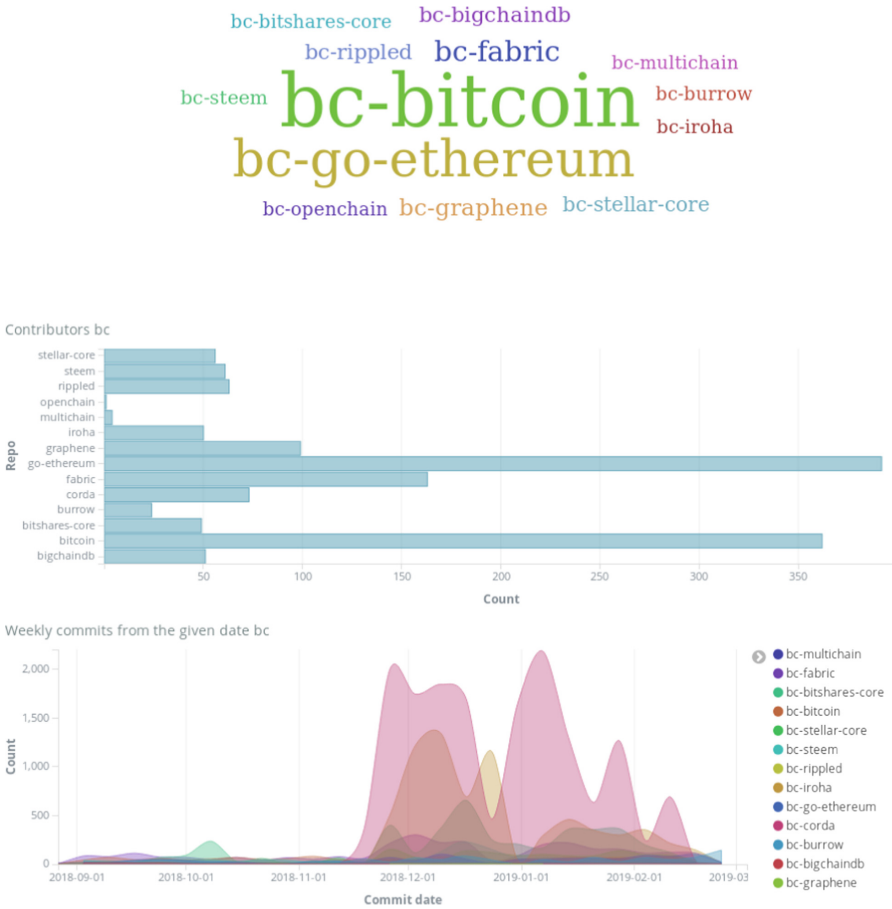

Fig. 4. Charts of comparison repositories. The top chart shows the total number of contributers. The bottom chart shows a weekly commit statistics. Tag cloud shows all repositories presented in database.

on all graphs will be shown exclusively. We have added a tag cloud with the names of all repositories for more convenient filtering.

The ability to filter the names of repositories of platforms of distributed ledgers in the tag cloud by the consensus algorithm has been added. Furthermore, one more cloud of names of consensus algorithms and an additional table of correspondences of algorithms and distributed ledger platforms has been added.

4.2 Kibana Graph

The Kibana Graph tool provides the opportunity to visualize dependencies between data in the elasticsearch database [8]. Unlike Kibana Visualization charts graphs cannot be loaded dynamically. Therefore, when updating the

database, one needs to update the graph manually. Nevertheless, in this way it is possible to present the information about dependencies between distributed ledger platforms in a simple and understandable form. Are some examples of the graph.

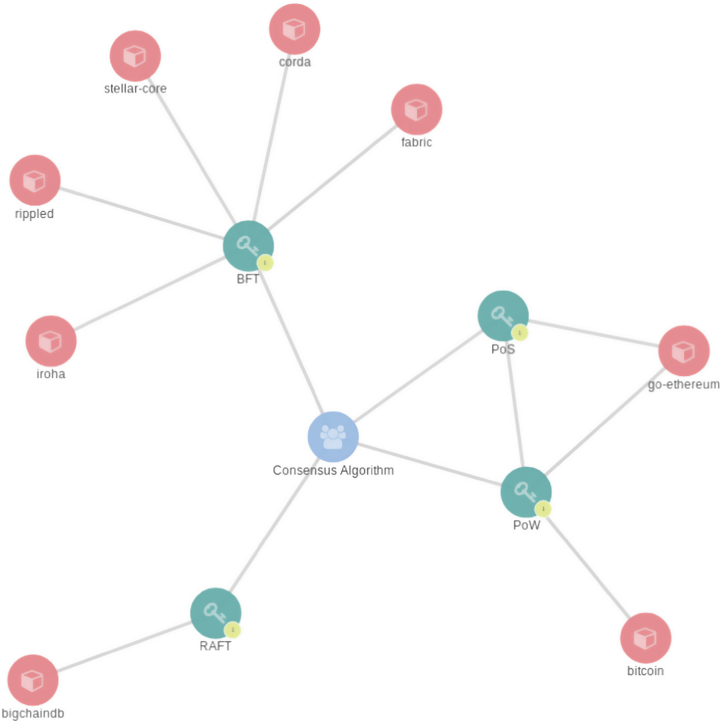

Fig. 5. Kibana Graph: consensus protocols and platforms which use them.

Figure 5 shows a graph with relations between consensus algorithms and the distributed ledger that are in use. In this figure, only four protocols are presented as an example for readability.

Figure 6 shows the dependencies between satellite repositories and their owners. Note that both, organization and user, can be an owner. If the repository is a fork of another one presented in the database, then the owner is referred to a specific user and not to an organization. In Fig. 6, such a relation is shown between the `tangaroa` repository and some users. Some of them made a fork of `juno` project as well.

In Fig. 7 the graph of top contributors for hyperledger satellite repositories is presented. Here, only those contributors have indicated the whose number of commits of the last 3 months in the corresponding repository exceeds 50.

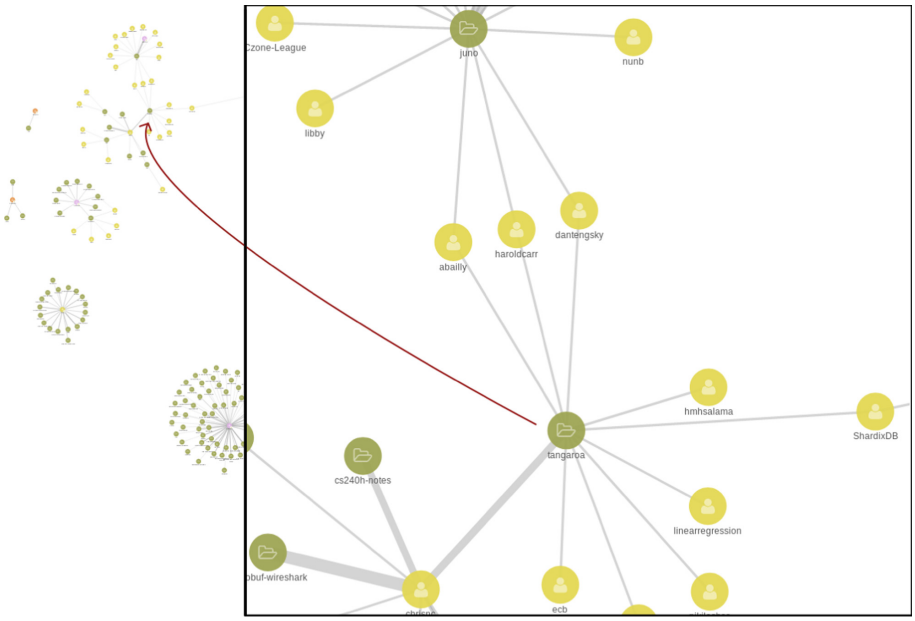

Fig. 6. Kibana Graph: satellite repositories and their owners.

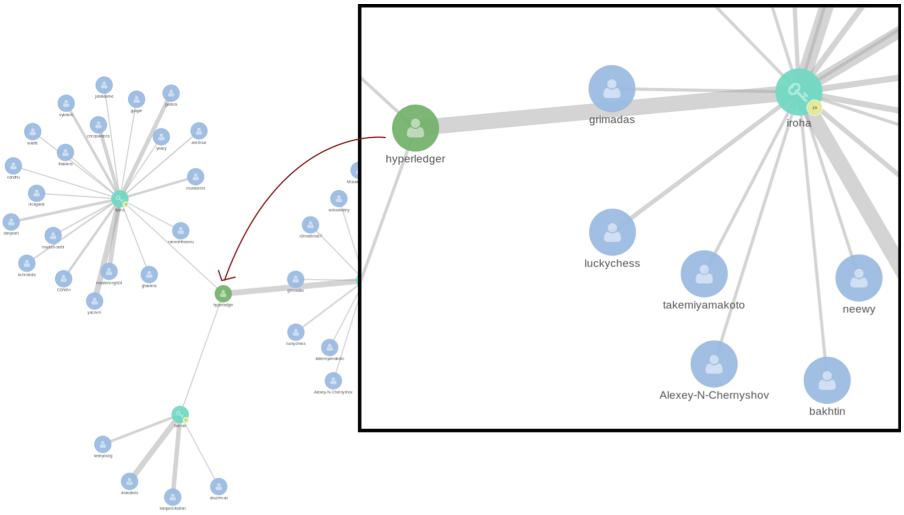

Fig. 7. Kibana Graph: top contributors of Hyperledger satellites.

5 Summary and Future Work

The system presented in this paper allows us to collect, update, visualize data on various distributed ledger technologies automatically, based on their open GitHub repositories. We also provide the opportunity to automatically evaluate and rank technologies according to the developed criteria. These criteria allow considering the activity of technology development, as well as the interaction of developers with users, interest in this technology and the dynamics of product development. Ledger evaluation occurs automatically. Note that this system can be generalized to the repositories of any subject.

Nevertheless, the developed system has a number of disadvantages. It is possible to evaluate only those repositories that are publicly available on GitHub. In addition, the construction of graphs is not automated and in the current implementation, it requires manual updating when updating the database.

When plotting, Kibana needs to run the database queries again. On the one hand, this allows seeing the actual data at any moment. However, on the other hand, it leads to a decrease in performance when the database grows. In the future, we plan to abandon Kibana Graph in favor of Neo4j, which has free community license or free-to-use GraphDB.

6 Conclusion

The presented system allows us to visually get up to date information on the development of projects based on their GitHub repositories, as well as to rank and evaluate them. The paper proposes criteria for evaluating repositories, developed on the basis of open data obtained using the GitHub API. It also describes the possibilities of visualization of this data. Most of the system works in automatic mode, which allows the system to scale to a much larger number of projects to be considered.

Acknowledgements. We are grateful to Dr. Martin Reinhardt for the help to prepare the text of this paper.

References

1. Salimitari, M., Chatterjee, M.: An overview of blockchain and consensus protocols for IoT networks. arXiv preprint arXiv:1809.05613 (2018)
2. Seibold, S., & Samman, G.: Consensus: immutable agreement for the internet of value. https://assets.kpmg.com/content/dam/kpmg/pdf/2016/06/kpmgblockchain-consensus-mechanism.pdf. Accessed 20 Mar 2019
3. Valenta, M., Sandner, P.: Comparison of ethereum, hyperledger fabric and corda. In: FSBC Working Paper, June 2017. Frankfurt School Blockchain Center (2017)
4. Macdonald, M., Liu-Thorrold, L., Julien, R.: The blockchain: a comparison of platforms and their uses beyond bitcoin. Working Paper, pp. 1–18 (2017)
5. Xu, X., et al.: A taxonomy of blockchain-based systems for architecture design. In: 2017 IEEE International Conference on Software Architecture (ICSA), pp. 243–252. IEEE (2017)

6. GitHub Rest API v3, Rate limiting. https://developer.github.com/v3/. Accessed 20 Mar 2019
7. Elastic. Kibana. https://www.elastic.co/products/kibana. Accessed 20 Mar 2019
8. Elastic. ElasticSearch. https://www.elastic.co/products/elasticsearch. Accessed 20 Mar 2019
9. Dinh, T.T.A., Wang, J., Chen, G., Liu, R., Ooi, B.C., Tan, K.L.: Blockbench: a framework for analyzing private blockchains. In: Proceedings of the 2017 ACM International Conference on Management of Data, pp. 1085–1100. ACM (2017)
10. Dinh, T.T.A., Liu, R., Zhang, M., Chen, G., Ooi, B.C., Wang, J.: Untangling blockchain: a data processing view of blockchain systems. IEEE Trans. Knowl. Data Eng. **30**(7), 1366–1385 (2018)