Lean GHTorrent: GitHub data on demand

Georgios Gousios[*], Bogdan Vasilescu[†], Alexander Serebrenik[†], Andy Zaidman[*]

[*]Delft University of Technology
Delft, The Netherlands
{g.gousios, a.e.zaidman}@tudelft.nl

[†]Eindhoven University of Technology
Eindhoven, The Netherlands
{b.n.vasilescu, a.serebrenik}@tue.nl

ABSTRACT

In recent years, GITHUB has become the largest code host in the world, with more than 5M developers collaborating across 10M repositories. Numerous popular open source projects (such as Ruby on Rails, Homebrew, Bootstrap, Django or JQuery) have chosen GITHUB as their host and have migrated their code base to it. GITHUB offers a tremendous research potential. For instance, it is a flagship for current open source development, a place for developers to showcase their expertise to peers or potential recruiters, and the platform where social coding features or pull requests emerged. However, GITHUB data is, to date, largely underexplored. To facilitate studies of GITHUB, we have created GHTorrent, a scalable, queriable, offline mirror of the data offered through the GITHUB REST API. In this paper we present a novel feature of GHTorrent designed to offer customisable data dumps on demand. The new GHTorrent data-on-demand service offers users the possibility to request via a web form up-to-date GHTorrent data dumps for any collection of GITHUB repositories. We hope that by offering customisable GHTorrent data dumps we will not only lower the "barrier for entry" even further for researchers interested in mining GITHUB data (thus encourage researchers to intensify their mining efforts), but also enhance the replicability of GITHUB studies (since a snapshot of the data on which the results were obtained can now easily accompany each study).

1. INTRODUCTION

During recent years, GITHUB (2008) has become the largest code host in the world, with more than 5M developers collaborating across 10M repositories. Due to its support for distributed version control (Git) and pull-based development [2], as well as its modern Web UI and focus on social coding [4], GITHUB has quickly surpassed in size and popularity even much older forges such as Sourceforge (1999). As a result, numerous projects (especially open source) are migrating their code base to GITHUB (for instance, the Google query *migrate to github* returns more than 4M results), which now hosts popular projects such as Ruby on Rails, Homebrew, Bootstrap, Django or jQuery.

Researchers have quickly jumped on board and have started exploring GITHUB data. So far, studies focused on building language topic models of source code [1], understanding the effects of branching and pull-based software development [9, 14], uncovering associations between crowdsourced knowledge and software development [21], visualizing collaboration and influence [12], exploring the social network of developers [19],or investigating how the social nature of GITHUB impacts collaboration and impression formation [4] and could be used to improve development practices [15]. More studies are expected to be published this year, since GITHUB is the topic of the Mining Challenge at the 2014 edition of the Working Conference on Mining Software Repositories (MSR). However, as opposed to Stack Overflow (also 2008), the largest Q&A site for programming-related questions and the topic of the Mining Challenge at the 2013 edition of MSR, the richness of GITHUB data remains largely underexplored in terms of academic publications [20].

To facilitate studies of GITHUB, we have created GHTorrent [10], a scalable, queriable, offline mirror of the data offered through the GITHUB REST API. GHTorrent data has already been used in empirical studies (e.g., [9, 18, 21]). In this paper we present a novel feature designed to offer customisable data dumps on demand. The new GHTorrent data-on-demand service offers users the possibility to request via a web form up-to-date GHTorrent data dumps (in both MySQL and MongoDB formats) for any collection of GITHUB repositories.

Apart from lowering the "barrier for entry" even further for researchers interested in mining GITHUB data, this data-on-demand service offers several advantages. Firstly, while the GHTorrent project already offered data dumps of both its raw data (MongoDB, currently more than 2TB) and metadata (MySQL, currently more than 20GB), downloading and restoring these dumps can be very time consuming and might not be necessary if a particular analysis is restricted in scope to say a handful of "interesting" GITHUB projects (e.g., the Ruby on Rails project, for which separate data sets also started being collected [23]).

Secondly, while the idea of running queries with a restricted scope is not necessarily new with respect to the official release of GHTorrent [10], the data-on-demand service enhances replicability of results obtained using GHTorrent data. GHTorrent already offered an online query interface with access to an archived version of the relational database, which could be used to restrict the scope of a query. However, GITHUB is a very dynamic platform where developers, projects and wikis are created and deleted constantly. Therefore, online queries of GHTorrent data may return different results at different times if project data recorded by GHTorrent has been refreshed in the meantime. To enhance the replicability [7] of such results, it is therefore preferable to store the exact snapshot of the data set used in the analysis.

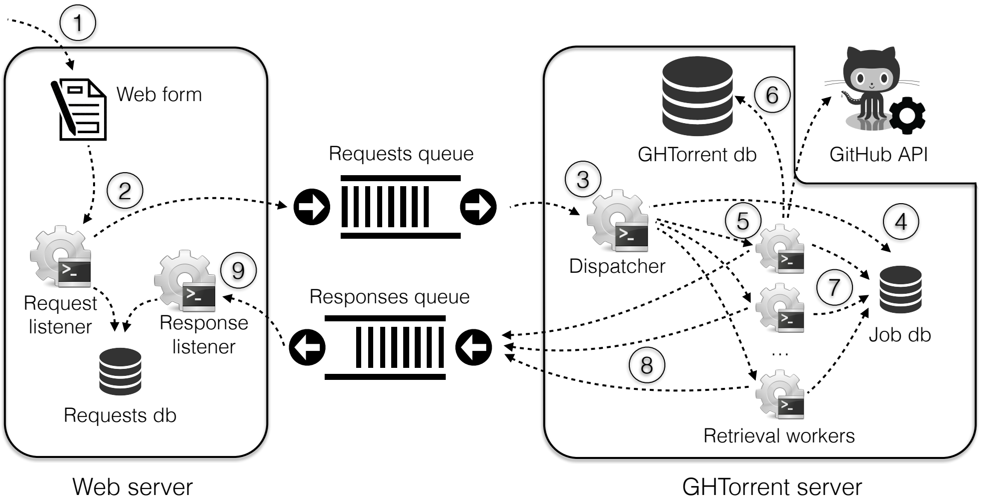

Figure 1: Architecture of the GHTorrent data-on-demand service.

Thirdly, our experiences with curating academic papers based on Stack Exchange data [20] suggest that researchers prefer to work with data dumps rather than online data explorers (for reasons such as eliminating the reliance on a third party service, replicability, or integration with existing tooling or infrastructure). Even after factoring out papers published at the Mining Challenge of MSR 2013, an overwhelming fraction of the remaining Stack Exchange papers published after 2010 (when the Stack Exchange data explorer became available) have used data dumps rather than the data explorer. We hope that by offering customisable GHTorrent data dumps we will encourage researchers to intensify their efforts to mine GitHub data.

The rest of this paper is organised as follows. In Section 2 we describe the architecture of lean GHTorrent, followed by a discussion of how to use the service in Section 3 and current limitations in Section 4. Next, we discuss related work in Section 5, and present our conclusions in Section 6.

2. ARCHITECTURE

The architecture of the new GHTorrent data-on-demand service consists of two loosely coupled parts: a web server that handles data requests from users and the GHTorrent server that performs the data extraction. The two servers communicate via messaging queues.

The interaction between the different subcomponents of the web and GHTorrent servers is illustrated in Figure 1. First, users specify their requests for data by filling in the web form at `http://ghtorrent.org/lean` ①. A request listener validates each request (e.g., it must contain an email address, it must not ask for more than 1000 repositories) and records metadata about its owner, payload, timestamp and status (completed, in progress) in a relational database. Then, for each GitHub repository part of the request, the listener posts a message to the queue ② containing the request identifier, timestamp and repository.

On the GHTorrent server side, a dispatcher listens in on the requests queue ③ and interprets the messages received as follows. First, if the message refers to a request for which no previous messages (asking for different repositories) have been received, a new shared relational database is created to collect metadata for the repositories part of this request ④. This database has the same schema as the original GHTorrent MySQL database [10], reproduced for completeness in Figure 2. Then, for each message (repository) referring to the same job, a retrieval worker is instantiated having as parameters the repository being requested, details for connecting to the job database, and the timestamp of the request ⑤.

Retrieval workers run in parallel and make extensive use of caching. If the main GHTorrent MongoDB database already contains data for this repository, then the shared job database is populated with metadata for this repository extracted from the main GHTorrent MongoDB database ⑥. Otherwise, both the main GHTorrent database and the job database ⑦ are updated with data freshly extracted from the GitHub API. This data collection process, again designed as a decentralized process, with decentralization mediated using a similar *worker queue model*, was described previously [8]. Once a retrieval worker finishes, it posts a message to the responses queue ⑧ signalling the completion of its task.

On the web server side, a response listener handles incoming response messages (one for each repository in each request) ⑨ and updates the status of the job in the requests database. When "task complete" messages have been received for all repositories part of a request, data dumps are being created from both the job database (MySQL, having the GHTorrent schema [8]) and the main GHTorrent database (MongoDB, only collections—groupings of MongoDB documents—relevant for this request are extracted).

Finally, the request owner is notified via email that her job has completed and the requested data dumps are available for download at a given URL.

3. USING THE SERVICE

Essentially any study of a restricted collection of GitHub repositories can be carried out using the lean GHTorrent offering the advantages of flexibility in selecting the repositories and reproducibility of the results.

We envision, for example, use cases in which researchers interested in mining GitHub data start off by using the in-browser interface to select a number of GitHub repositories matching their research goals. Then, lean GHTorrent can be used to retrieve data for those repositories.

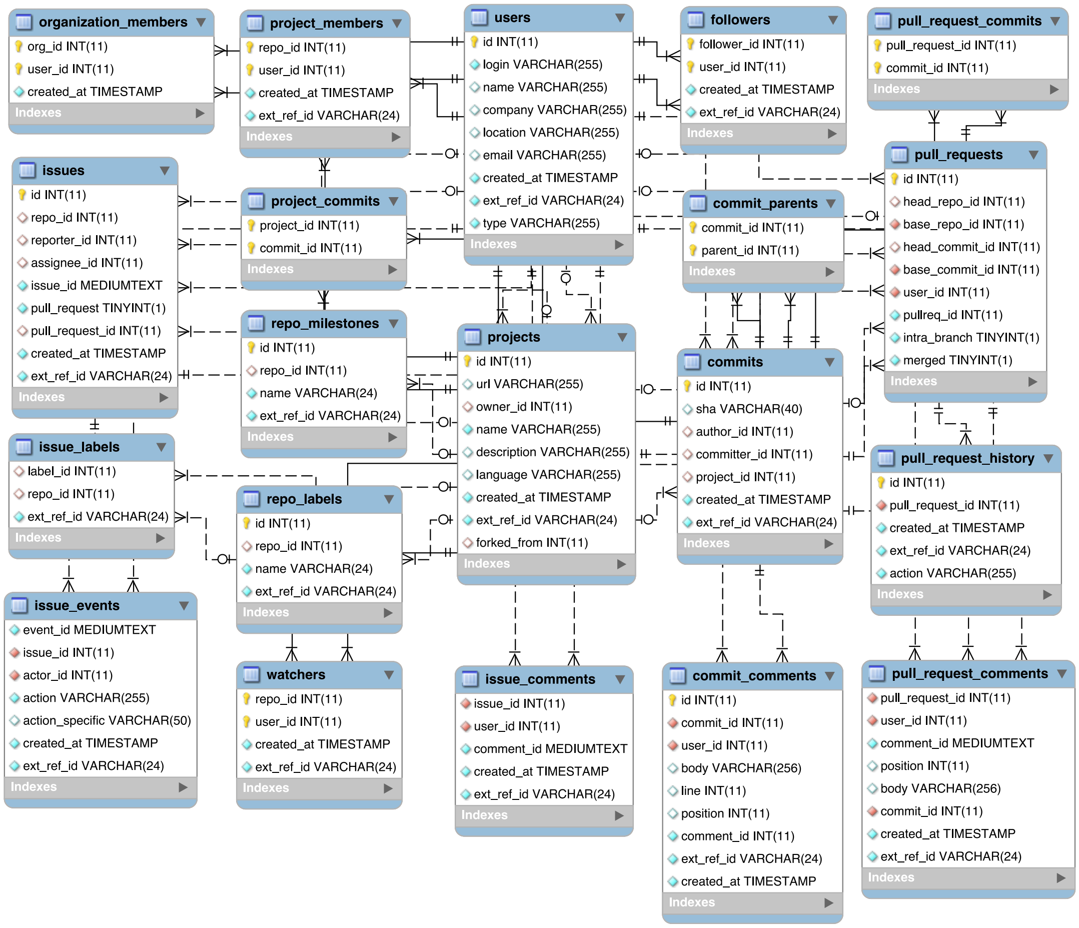

Figure 2: MySQL database schema [8].

To use the web form at `http://ghtorrent.org/lean`, repositories should be input one per line in the dedicated space. The input format for a repository is <*owner*>/<*repository*> (for instance, *gousiosg/github-mirror*, or *rails/rails*). To select the repositories that are interesting for analysis, researchers can use the existing GHTorrent MySQL web interface for filtering projects based on specific criteria (e.g. all Ruby on Rails forks, projects in Java that have more than 100 pull requests, projects that received a commit throught a pull request in 2014 etc).

Once a job has been submitted, the user is sent an email with a tracking URL, where information about the status of retrieving each component (table; commits, forks, pull requests, project members, etc.) of each requested repository is displayed. Refreshing the tracking page will update the status information.

Once the job finishes, an archive containing the MySQL and MongoDB data dumps is offered for download. The MySQL dump contains metadata for the requested repositories, having the schema described in Figure 2. The MongoDB dump contains all the data extracted by GHTorrent from the GITHUB API (e.g., in addition to metadata about users, organisations, repositories, commits, issues and pull requests already available in MySQL, it contains the actual changes—diffs—to the repository for each commit). To restore the database dumps locally, the standard procedure of importing `sql` archives or `mongo` collections (i.e., using the `mongorestore` script provided with MongoDB) applies.

Furthermore, the Ruby scripts provided together with GHTorrent (e.g., scripts to update all the data related to a given repository or a given user; see the GHTorrent GITHUB repository `https://github.com/gousiosg/github-mirror`) allow users of lean GHTorrent, once they restore locally the database dumps they requested, to update their local copies independently.

4. LIMITATIONS

Designing and implementing a robust and scalable lean GHTorrent service was a challenging endeavour. Still, in its current implementation lean GHTorrent has a number of limitations. First, dumps contain only the first order dependencies (e.g., contributors to a repository and their followers, but not followers of these followers). Second, depending on the size of the request and the load on GHTorrent servers at that time, creating the dumps can be a lengthy process, potentially requiring several days to complete. Third, no recovery actions in case of errors are currently implemented, potentially leading to incomplete dumps, e.g., if GITHUB fails to answer an API request. Researchers using lean GHTorrent data are advised to check the integrity of the data dumps themselves and, in case of incomplete data, use the `ght-retrieve-*` scripts in the main GHTorrent distribution to fill in the data holes. Finally, to limit the load on GHTorrent servers, requests to lean GHTorrent should not exceed 1000 repositories.

5. RELATED WORK

The idea of providing or retrieving software repository data on-demand as such is not new and can be seen as related to “Data as a Service” or “Information as a Service” [5]. The data being provided was usually limited to the meta-data [3] or elements of the repository such as files [22]. Similarly to the latter work, lean GHTorrent provides elements of GITHUB. However, GITHUB is a *repository of repositories* [17] or *meta repository* [11] and, therefore, its elements are repositories themselves. Meta repositories, including lean GHTorrent, provide for cross-domain analysis [17]. As opposed to existing meta repositories such as OHLOH or FLOSSMOLE [13], lean GHTorrent provides the researchers with the possibility to select their own object of study rather than being forced to analyse the entire collection searching for the proverbial needle.Moreover, as opposed to such effort as Boa [6] integrating the repository analysis tasks in the web-based interface, lean GHTorrent allows the researchers to download the relevant repositories and subject them to further processing by independent tools, *i.e.*, the analysis tasks are not restricted to the functionality provided the web-interface.

Projects hosted GITHUB or the entire GITHUB collection have been subject to numerous studies (e.g., [1,4,9,12,14–16,19]). The GITHUB mirror and the predecessor of the current work [10], has been used in empirical studies [18,21].

6. CONCLUSIONS

We presented a novel feature of GHTorrent that allows users to request GITHUB data dumps on demand for any collection of GITHUB projects (repositories). Lean GHTorrent offers several advantages, being lightweight and easy to use, fostering replicability and offering flexibility and independence to researchers interested in mining GITHUB. Together with the existing GHTorrent infrastructure, the new lean data-on-demand service lowers the “barrier for entry” for GITHUB miners to a minimum. We hope this will encourage researchers to intensify their efforts to mine GITHUB data, as well as serve as inspiration for others willing to share software engineering datasets (the implementations of both GHTorrent and lean GHTorrent are publicly available).

Acknowledgements. Georgios Gousios is funded throught the NWO TestRoots project (639.022.314). Bogdan Vasilescu has been supported by the research project NWO 600.065.120.10N235 financed by Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO).

7. REFERENCES

[1] M. Allamanis and C. Sutton. Mining source code repositories at massive scale using language modeling. In *MSR*, pages 207–216. IEEE, 2013.

[2] E. T. Barr, C. Bird, P. C. Rigby, A. Hindle, D. M. German, and P. Devanbu. Cohesive and isolated development with branches. In *FASE*, pages 316–331. Springer, 2012.

[3] A. Begel, Y. P. Khoo, and T. Zimmermann. Codebook: discovering and exploiting relationships in software repositories. In *ICSE*, volume 1, pages 125–134, May 2010.

[4] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in Github: transparency and collaboration in an open software repository. In *CSCW*, pages 1277–1286. ACM, 2012.

[5] A. Dan, R. Johnson, and A. Arsanjani. Information as a service: Modeling and realization. In *International Workshop on Systems Development in SOA Environments*, page 2, May 2007.

[6] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In *Proceedings of the 35th International Conference on Software Engineering*, pages 422–431, May 2013.

[7] J. M. González-Barahona and G. Robles. On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Empirical Software Engineering*, 17(1-2):75–89, 2012.

[8] G. Gousios. The GHTorent dataset and tool suite. In *MSR*, pages 233–236. IEEE, 2013.

[9] G. Gousios, M. Pinzger, and A. van Deursen. An exploratory study of the pull-based software development model. In *ICSE*. ACM, 2014.

[10] G. Gousios and D. Spinellis. GHTorrent: Github’s data from a firehose. In *MSR*, pages 12–21. IEEE, 2012.

[11] V. Gruhn, C. Hannebauer, and C. John. Security of public continuous integration services. In *9th International Symposium on Open Collaboration*, WikiSym ’13, pages 15:1–15:10, New York, NY, USA, 2013. ACM.

[12] B. Heller, E. Marschner, E. Rosenfeld, and J. Heer. Visualizing collaboration and influence in the open-source software community. In *MSR*, pages 223–226. ACM, 2011.

[13] J. Howison, M. Conklin, and K. Crowston. FLOSSmole: A collaborative repository for FLOSS research data and analyses. *International Journal of Information Technology and Web Engineering*, 1(3):17–26, 2006.

[14] H. Lee, B.-K. Seo, and E. Seo. A git source repository analysis tool based on a novel branch-oriented approach. In *ICISA*, pages 1–4. IEEE, 2013.

[15] R. Pham, L. Singer, and K. Schneider. Building test suites in social coding sites by leveraging drive-by commits. In *ICSE*, pages 1209–1212. IEEE, 2013.

[16] D. Pletea, B. Vasilescu, and A. Serebrenik. Security discussions on GitHub: Topic mining and sentiment analysis. In *MSR*. IEEE, 2014.

[17] S. K. Sowe, L. Angelis, I. Stamelos, and Y. Manolopoulos. Using repository of repositories (RoRs) to study the growth of F/OSS projects: A meta-analysis research approach. In J. Feller, B. Fitzgerald, W. Scacchi, and A. Sillitti, editors, *Open Source Development, Adoption and Innovation*, volume 234 of *IFIP—The International Federation for Information Processing*, pages 147–160. Springer US, 2007.

[18] M. Squire. Forge++: The changing landscape of FLOSS development. In *HICSS47*. IEEE, 2014.

[19] F. Thung, T. F. Bissyandé, D. Lo, and L. Jiang. Network structure of social coding in GitHub. In *CSMR*, pages 323–326. IEEE, 2013.

[20] B. Vasilescu. Academic papers using Stack Overflow data. `http://meta.stackoverflow.com/q/134495`, 2012.

[21] B. Vasilescu, V. Filkov, and A. Serebrenik. StackOverflow and GitHub: associations between software development and crowdsourced knowledge. In *SocialCom*, pages 188–195. IEEE, 2013.

[22] L. Voinea and A. Telea. Mining software repositories with CVSgrab. In *MSR*, MSR ’06, pages 167–168, New York, NY, USA, 2006. ACM.

[23] P. Wagstrom, C. Jergensen, and A. Sarma. A network of rails: a graph dataset of ruby on rails and associated projects. In *MSR*, pages 229–232. IEEE, 2013.