# Liquid Benchmarking: A Platform for Democratizing the Performance Evaluation Process

Sherif Sakr [†,‡], Amin Shafaat [†], Fuad Bajaber [*]
Ahmed Barnawi [*], Omar Batarfi [*], Abdulrahman Altalhi [*]
[†]University of New South Wales, Sydney, Australia
{ssakr,ashafaat}@cse.unsw.edu.au
[*]King Abdulaziz University, Jeddah, Saudi Arabia
{fbajaber,ambarnawi,obatarfi,ahaltalhi}@kau.edu.sa
[‡]King Saud bin Abdulaziz University for Health Sciences, National Guard, Riyadh, Saudi Arabia

## ABSTRACT

Performances evaluation, reproducibility and benchmarking represent crucial aspects for assessing the practical impact of research results in the computer science field. In spite of all the benefits (e.g., increasing impact, increasing visibility, improving the research quality) that can be gained from performing extensive experimental evaluation or providing reproducible software artifacts and detailed description of experimental setup, the required effort for achieving these goals remains prohibitive. In practice, conducting an independent, consistent and comprehensive performance evaluation and benchmarking is a very time and resource consuming process. As a result, the quality of published experimental results is usually limited and constrained by several factors such as: limited human power, limited time, or shortage of computing resources.

We demonstrate *Liquid Benchmarking* as an online and cloud-based platform for democratizing the performance evaluation and benchmarking processes. In particular, the platform facilitates the process of sharing the experimental artifacts (software implementations, datasets, computing resources, benchmarking tasks) as services where the end user can easily create, mashup, run the experiments and visualize the experimental results with zero installation or configuration efforts. In addition, the collaborative features of the platform enables the user to share and comment on the results of the conducted experiments so that it can guarantee a transparent scientific crediting process. Furthermore, we demonstrate four benchmarking case studies that have been implemented using the Liquid Benchmarking platform on the following domains: XML compression techniques, graph indexing and querying techniques and string similarity join algorithms.



## 1. INTRODUCTION

The last two decades have seen significant growth in the number of scientific research publications. One of the distinguishing characteristic of computer science research is that it produces *artifacts* in addition to the scientific publications, in particular *software implementations*. In general, reproducibility of experimental results represents a cornerstone in the computer science research field [2]. In practice, several benefits can be gained from providing reproducible experimental results including the improvement of the research quality, the gain of scientific credibility in addition to increasing the research visibility and the impact [2]. In particular, in an ideal world of computer science research, researchers describe the core of their contributions in the paper and then publicly provide the experimental datasets and the source codes/binaries of their software implementation for the community in order to facilitate the reproducibility of the published results in their publication. However, the world is not always ideal. While most of the computer science research literature usually present experimental results that evaluate/compare their proposed scientific contributions, the quality of such experimental results are usually limited due to several factors including: insufficient effort or time, unavailability of suitable test cases or any other resource constraints [8]. Furthermore, researchers used to focus on reporting about the *sweet spots* of their work in a way that is usually do not cover the ultimate picture or the practical insights of the real-world scenarios or the different application domains.

In principle, conducting an *independent* and *comprehensive* benchmarking study for the-state-of-the-art in a certain research topic is usually a very useful but a very challenging task as well. In particular, it usually consumes a lot of time and efforts due to multiple factors such as: unavailability of standard benchmarking tasks, lack of access to the implementations (source code or binaries) for some techniques which are proposed in the research literature in addition to the constraints of getting an access to different configuration of computing resources/environments that reflect the wide spectrum of different real-world scenarios [8]. Therefore, it is, unfortunately, quite common in several research domains to have no or little objective knowledge regarding the pros and cons of any set of different proposed research approaches/techniques which are sharing the goal of tackling a specific research challenge.


10.5441/002/edbt.2015.52

Recently, the challenge of defining and conducting comprehensive performance evaluations and benchmarking studies has been recognized by different research communities. In addition, several conferences, publishers and funding agencies have started to encourage their authors to provide the descriptions and the software artifacts that can facilitate the reproducibility of the experimental results of their publications. For example, in the database research community, ACM SIGMOD 2008 was the first conference that offered to verify the repeatability of the published experiments by allowing the authors to submit their programs and experimental datasets [6]. In addition, since 2008, the VLDB conference has created a new experimental and analysis track that encourages the research community to publish manuscripts that report and document thorough experimental evaluation and benchmarking studies[1]. Furthermore, several proposals [1] and tutorials have been presented in the major database venues to promote the crucial importance of performance evaluation, reproducibility and benchmarking in database research [2, 5]. Other research communities have been following the same approach such as the Semantic Web[2,3,4], Semantic Web Service[5], Business Process[6], Information Retrieval[7] in addition to the general Executable Paper Grand Challenge[8]. Although such types of research publications and benchmarking efforts are useful and important, however, they suffer from a main limitation which is that they present particular *snapshots* for the state-of-the-art that reflect the status at the time of their execution. In practice, the state-of-the-art in any research domain is always *dynamic* and *evolving* by default. For instance, new techniques that address the same research challenge of a previously published snapshot paper can be introduced or the performance characteristics of previously evaluated techniques can differ. Thus, such papers can be outdated shortly after they have been published.

In this paper, we demonstrate *Liquid Benchmarking* [8] as an online, collaborative and cloud-based platform that seeks to remedy the above mentioned challenges and problems by facilitating the *democratization* and improving the quality of the performance evaluation and benchmarking processes in the computer science research domain. In particular, we summarize the main strengths of our platform as follows:

- The platform dramatically reduces the time and effort for conducting performance evaluation process by facilitating the process of sharing the experimental artifacts (software implementations, datasets, computing resources, benchmarking tasks) and enabling the users to easily create, mashup and run the experiments with zero installation or configuration efforts.
- The platform supports for searching, comparing, analyzing and visualizing (using different built-in visualization tools) the results of previous experiments.
- The users of the platform can subscribe to get notifications about the results of any new running experiments for the domains/benchmarks of their own interests.
- The social and collaborative features of the platform enables turning the performance evaluation and benchmarking process into a *living* process where different users can run different experiments, share the results of their experiments with other users in addition to commenting on the results of the conducted experiments by themselves or by other users of the platform. Such features guarantee the utilization of the *wisdom of the crowd*, the *freshness* of the results, the establishment of a *transparent* process for scientific *crediting* and the development of scientific advances that trust and build on previous research contributions.

[1] http://www.vldb.org/pvldb/vol1.html
[2] http://challenge.semanticweb.org/
[3] http://2014.eswc-conferences.org/important-dates/call-challenges
[4] http://iswc2014.semanticweb.org/call-replication-benchmark-data-software-papers
[5] http://sws-challenge.org/wiki/index.php/Main\textunderscorePage
[6] http://processcollections.org/past/2013-2/matching-contest
[7] http://www2.informatik.hu-berlin.de/~wandelt/searchjoincompetition2013/
[8] http://www.executablepapers.com/

# 2. PLATFORM DESIGN

## 2.1 Underlying Technologies

The features and design decisions of the Liquid Benchmarking platform combine the facilities provided by different technologies as follows:

- *Software-as-a-Service*: The platform relies on the RESTful architectural style as an effective software distribution mechanism in which software implementations get hosted on the computing environments and made available as web services to the end-users over the Internet. Such mechanism requires zero downloading, installation or configuration effort at the side of the end user where all communication with software can be achieved using HTTP methods.
- *Cloud Computing*: The platform utilizes cloud computing as an effective technology for broad sharing of hardware resources and computing environments via the Internet. In particular, virtualization is a key technology of the cloud computing paradigm that improves the manageability of hardware resources by flexibly allowing computing resources to be provisioned on demand (in the form of virtual machines) and hiding the complexity of resource sharing details from cloud users. In practice, conducting a fair and *apples-to-apples* comparison between any competing software implementations requires performing their experiments using *exactly* the same computing environment [8]. In addition, performing a comprehensive and insightful evaluation process that assess different performance characteristics of the evaluated software implementations may require using several virtual machines with variant and scaling (in terms of computing resources) configuration settings (e.g. main memory, disk storage, CPU speed) that reflect different real-world scenarios [8]. The Liquid Benchmarking platform utilizes the virtualization technology for maintaining the testing computing environments in cloud platforms in the form of pre-configured *virtual machines* (with different configurations) which are hosting the competing software implementations (in the form of web services) and are shared by the end-users of the benchmark.
- *Collaborative and Social Software*: The platform is enabled with different Web 2.0 capabilities (e.g. user comments, tagging, forums) that support human inter-

action and facilitates the building of online communities between groups of researchers who share the same interests (peers) where they can interact and work together in an effective and productive way. Most important, the platform supports sharing the performance evaluation and benchmarking artifacts (e.g., software implementations, datasets, virtual machines) in a *workable* environment.

## 2.2 Benchmark Specifications

In Liquid Benchmarking, each benchmark is configured by defining the following main components:

- *Evaluated Solutions*: Represent the set of competing software implementations (e.g. algorithms, techniques, systems) which are sharing the goal of tackling the subject research challenge of the benchmark. The implementation of each evaluated solution needs to be wrapped with a web service interface before being integrated on the benchmark.
- *Service Schema*: Defines the set of parameters (inputs and outputs) that need to be defined for interfacing with the services of the evaluated solutions.
- *Task(s)*: Describes an operation which is specified for evaluating the competing implementations (e.g. queries, update operations, compression operations). In particular, each task represent an *instantiation* for the parameters of the service schema with a set of value that describes the specification of the task.
- *Metric(s)*: Represents a measure (e.g. execution time, response time, throughput) for evaluating the competing software implementations in performing the benchmarking tasks. In particular, it provides the basis for comparing the competing software implementations.
- *Testing Environment(s)*: Represents a set of resources configuration (e.g., CPU, disk, memory) for a computing environment (virtual machine) that hosts the services of competing software implementations.

## 2.3 Platform Components and Architecture

Figure 1 illustrates the architecture of the Liquid Benchmarks platform which are equipped with several *components* that are described as follows:

- **Web-based User Interface**: This component provides the end user with a user-friendly interface where he/she can *mash up* the components (e.g., services, tasks, metrics, computing environments) of the experiment in a *drag and drop* style. It also provides the end-user with other features such as: managing user account, maintaining the metadata store, searching and commenting on the results of previous experiments, subscribing to the results of a benchmark in addition to analyzing and visualizing the experimental results.
- **Metadata Store**: This component stores the information about the components (e.g., services, service schema, tasks, virtual machines) of the benchmark.
- **Experiment Manager**: The experiment manager receives the specification of the user-defined experiment, configured by the Liquid Benchmark UI, which is then registered for execution on the **Experiment Queue**. In principle, the experiment queue is used by the **Experiment Execution Engine** to ensure that the execution of one experiment in a testing environment is not going to influence the execution of another experiment in the same environment (an experiment can only start after the end of the current experiment, if exist, on the computing environment). Through the experiment life cycle, the **Experiment Execution Engine** sends a set of *notification events* to the **Notification Center** with the status of the experiment till its completion and storing its results in the **Repository of Experimental Results** for further analysis and visualization purposes. It should be noted that the **Experiment Execution Engine** is the component that is responsible for managing the life cycle of testing environments. In particular, it starts the virtual machine of a testing environment for running an experiment if it has been in a stopped mode or it stops the virtual machine if it has been idle for a while and has no pending experiments in the queue.

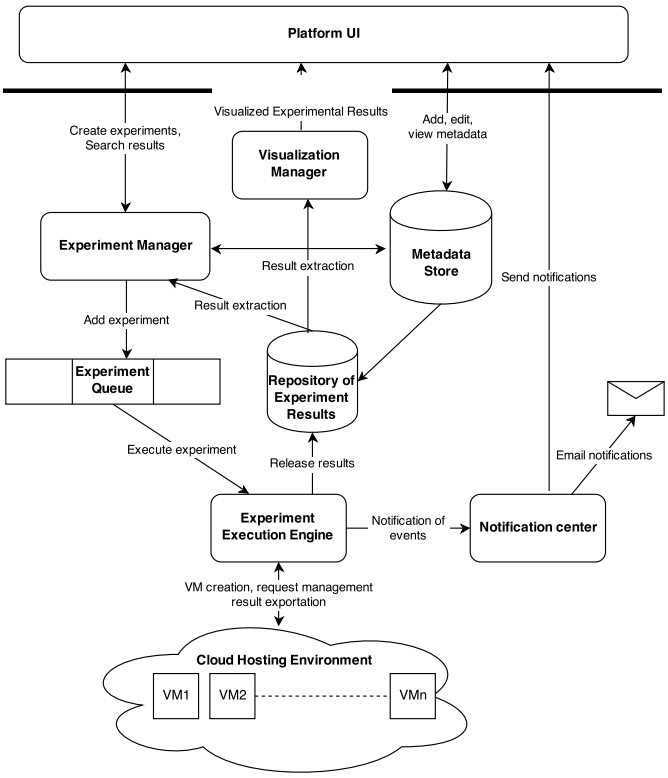


**Figure 1: Platfrom Architecture**

- **Repository of Experiment Results**: This repository stores the results of all experiments associated with their configuration parameters, *provenance* information (e.g. timestamp, user) and social information (e.g. comments, discussions). Clearly, end-users can search and view the contents of this repository to analyze, compare, visualize and comment on the results of the previously running experiments without taking the time of re-running or creating them from scratch.
- **Visualization Manager**: This component is equipped with a set of *visualization styles* (e.g. column charts, line charts) for presenting and comparing the results (metrics) of the selected experiments by the end-user.

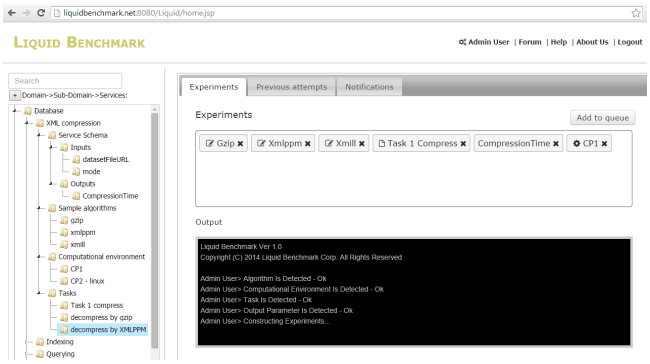


**Figure 2: Screenshot: Mashing Up an Experiment**

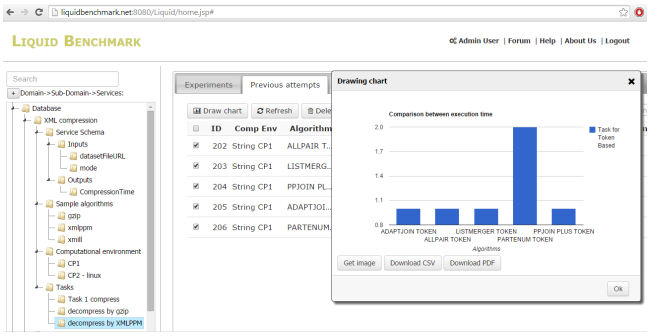


**Figure 3: Screenshot: Comparing and Visualizing Experimental Results**

## 3. DEMONSTRATION SCENARIOS

In this demonstration, we will start by presenting the different features of the Liquid Benchmarking platform[9] such as the process of mashing up a new experiment (Figure 2) or visualizing the experimental results (Figure 3). Then, the demonstration will present four benchmarking case studies that have been implemented using the Liquid Benchmarking platform on the following domains:

- *XML compression*[10]: This case study is based on the benchmark of XML compressors that has been presented in [7]. In particular, this case study provides services for the implementation of nine XML compression tools with benchmarking tasks over an XML corpus that contains 57 documents which are covering the different types and scales of XML documents. This case study evaluates the XML compressors by three different metrics: compression ratio, compression time and decompression time.
- *Graph indexing and querying*[11]: This case study implements the *iGraph* framework [3] for evaluating the graph indexing and querying techniques. In particular, the case study provides the services of seven techniques and evaluates them on the basis of their indexing time, index size and query processing time using a real AIDS antiviral screen dataset (NCI/NIH) and synthetically generated datasets.
- *String Similarity Join*[12]: An implementation for the recent evaluation and comparison study which is presented by Jiang et al. [4]. The case study provides the implementation of twelve algorithms and provides six different experimental datasets. The evaluation of the benchmarked algorithms is based on two metrics: the running time and the size of candidate results.

The case studies of our demonstration will be deployed in two cloud environments: the Amazon public cloud environment[13] in addition to our own private cloud environment which is managed by the *OpenStack* platform[14]. In addition, each case study will be demonstrated using two different testing environments (virtual machines): The first environment will be configured with high computing resources while the other environment will be conifgured with limited computing resources. Furthermore, we will present how the authenticated users can access different services of the platform (e.g., creating and running experiments, searching the repository of results) using its supported RESTful interfaces and API-based SDK[15].

[9]The platform can be accessed online on `http://liquidbenchmark.net:8080/Liquid/`. The full documentation for using the platform is available on `http://wiki.liquidbenchmark.net/`

[10]`http://wiki.liquidbenchmark.net/doku.php/casestudy-xmlcompression`

[11]`http://wiki.liquidbenchmark.net/doku.php/casestudy-graph-indexing-querying`

[12]`http://wiki.liquidbenchmark.net/doku.php/casestudy-string-similarity-join`

[13]`http://aws.amazon.com/`

[14]`http://www.openstack.org/`

[15]`http://wiki.liquidbenchmark.net/doku.php/RESTful-interface`

## Acknowledgement

The Liquid Benchmark Project is supported by King Abdulaziz City for Science and Technology (KACST), project 11-INF1992-03.

## 4. REFERENCES

[1] F. Chirigati, M. Troyer, D. Shasha, and J. Freire. A Computational Reproducibility Benchmark. *IEEE Data Eng. Bull.*, 36(4), 2013.

[2] J. Freire, P. Bonnet, and D. Shasha. Computational reproducibility: state-of-the-art, challenges, and database research opportunities. In *SIGMOD*, 2012.

[3] W. Han, J. Lee, M. Pham, and J. Xu Yu. iGraph: A Framework for Comparisons of Disk-Based Graph Indexing Techniques. *PVLDB*, 3(1), 2010.

[4] Y. Jiang, G. Li, J. Feng, and W. Li. String Similarity Joins: An Experimental Evaluation. *PVLDB*, 7(8), 2014.

[5] S. Manegold and I. Manolescu. Performance evaluation in database research: principles and experience. In *EDBT*, 2009.

[6] I. Manolescu, L. Afanasiev, A. Arion, J. Dittrich, S. Manegold, N. Polyzotis, K. Schnaitter, P. Senellart, S. Zoupanos, and D. Shasha. The repeatability experiment of SIGMOD 2008. *SIGMOD Record*, 37(1), 2008.

[7] S. Sakr. XML compression techniques: A survey and comparison. *JCSS*, 75(5):303–322, 2009.

[8] S. Sakr and F. Casati. Liquid Benchmarks: Towards An Online Platform for Collaborative Assessment of Computer Science Research Results. In *TPCTC*, 2010.