A Cloud-Based Platform for Democratizing and Socializing the Benchmarking Process

Fuad Bajaber
King Abdulaziz University
Jeddah, Saudi Arabia

Amin Shafaat
University of New South Wales
Sydney, Australia

Omar Batarfi
King Abdulaziz University
Jeddah, Saudi Arabia

Radwa Elshawi
Princess Nourah Bint Abdulrahman University
Riyadh, Saudi Arabia

Abdulrahman Altalhi
King Abdulaziz University
Jeddah, Saudi Arabia

Ahmed Barnawi
King Abdulaziz University
Jeddah, Saudi Arabia

Sherif Sakr
King Saud bin Abdulaziz University for Health Sciences, Riyadh, Saudi Arabia
University of New South Wales, Sydney, Australia

***Abstract*—Performances evaluation, benchmarking and reproducibility represent significant aspects for evaluating the practical impact of scientific research outcomes in the Computer Science field. In spite of all the benefits (e.g., increasing visibility, boosting impact, improving the research quality) which can be obtained from conducting comprehensive and extensive experimental evaluations or providing reproducible software artifacts and detailed description of experimental setup, the required effort for achieving these goals remains prohibitive. In this article, we present the design and the implementation details of the *Liquid Benchmarking* platform as a social and cloud-based platform for democratizing and socializing the software benchmarking processes. Particularly, the platform facilitates the process of sharing the experimental artifacts (computing resources, datasets, software implementations, benchmarking tasks) as services where the end users can easily design, mashup, execute the experiments and visualize the experimental results with zero installation or configuration efforts. Moreover, the social features of the platform enable the users to share and provide feedback on the results of the executed experiments in a form that can guarantee a transparent scientific crediting process. Finally, we present four benchmarking case studies that have been realized via the Liquid Benchmarking platform in the following domains: XML compression techniques, graph indexing and querying techniques, string similarity join algorithms and reverse K nearest neighbors algorithms.**

Keywords*—Cloud Computing; Benchmarking; Software-as-a-Service, Social Computing

I. Introduction

Over the last decades, the scientific community has been witnessing a significant increase in the amount of research outlets. In general, one of the important characteristic of the Computer Science research field is that the research outcomes provides artifacts other than the research outlets, in particular, computer software. In principle, the Computer Science scientific community is continuously witnessing claims on performance improvement from the various researchers and publications which have called for the necessity of conducting comprehensive and reproducible experimental assessments and comparisons between *competing alternative* software of approaches, algorithms or entire systems with the objective of evaluating the significance or the practical impact of the reported research contributions. In practice, most of the research outlets usually report results of their experimental evaluation to assess/compare their introduced scientific contributions with the state-of-the-art, however, unfortunately, the accuracy and the quality of such experimental evaluations are usually constrained with various factors including the unavailability of sufficient time or manpower, the unavailability of adequate or standard testing scenarios or any other resource constraints. In addition, it is common that research outlets are usually concentrating on reporting the experimental results of the *sweet spots* of their contribution which can usually affect on the reflection of the actual picture on the real-world scenarios and suffer from file-drawer effect [37]. Furthermore, it is usually very challenging to assess and understand the performance characteristics of the design choices of a specific approach.

Practically, conducting a consistent, independent and comprehensive study of performance evaluation or benchmarking competing alternatives in a specific domain is mostly a resource and time consuming process. Therefore, it is common that the accuracy and the quality of reported experimental results can be limited and constrained with various conditions including the limited time, limited human power, shortage of computing resources and unavailability of publicly accessible software implementation of some contributions that have been reported in the research literature. Moreover, it is practically challenging to get an access to various configuration of computing environments/resources which can represent or cover the wide spectrum of various real-world use cases [31]. Hence, it is, unfortunately, common in many research areas to have little or no objective knowledge about the advantages and limitations of any group of competing research approaches/techniques that are sharing the focus on addressing a specific research problem.

In principle, the ability to repeat experiments is considered a hallmark of the scientific process which is used to confirm or refute hypotheses and previously obtained results [16]. In recent years, the importance of defining and performing comprehensive benchmarking and performance evaluation studies has been acknowledged by various research communities. Additionally, various scientific conferences, funding agencies and publishers have started to motivate the researchers to share the software artifacts and documentations that can facilitate the reproducibility of the experimental results which are reported in their research outlets. For instance, in the database research community, since 2008, the ACM SIGMOD conference, which is considered as the most prestigious conference of the community, has started to offer the chance to verify the reproducibility of the reported experimental results by providing the researchers with the opportunity to submit their software and other experimental artifacts (e.g., datasets) [27]. Moreover, since 2008, another prestigious conference of the commuity, the VLDB conference, initiated a new experimental and analysis paper track that motivates the researchers of community to submit manuscripts that document and report in-depth experimental evaluation and benchmarking studies[1]. Furthermore, various proposals [10] and scientific tutorials demonstrated in the main scientific venues of the database research community have been focusing on promoting the significant importance of reproducibility, performance evaluation and benchmarking studies in database research [18], [25]. As a result, some efforts such as Arizona Database Laboratory (AZDBLab) [41] has been proposed to support database researchers in performing a comprehensive and empirical study among various database management systems. Other research communities followed the same trend such as the Semantic Web[2,3,4], Semantic Web Service[5], Business Process[6], Information Retrieval[7] communities in addition to the general Executable Paper Grand Challenge[8]. In spite the fact that these initiatives for benchmarking efforts and research publications are important and useful, however, the main limitation of these efforts is that they report particular *snapshots* for the state-of-the-art that reflect the status at the time of their conduct. In practice, the state-of-the-art in any research domain is always *evolving* and *dynamic*, by default. For example, emerging or novel approaches or techniques that tackle the same research challenge of a formerly revealed snapshot publication can be proposed or the performance characteristics of formerly assessed approaches or techniques may develop and improve. Hence, this type of research publications can be outdated shortly after they have been released.

In practice, the recent advances in Web technologies (e.g. social web, cloud computing, software-as-service) have provided novel work environments that opened new opportunities to address the above mentioned challenges. As a result, recently, the scientific communities started to increasingly use personal/shared blogs and wikis (e.g. ACM SIGMOD blog[9], DBMS2[10], SemWebTec[11]) to share and discuss their findings. *PubZone*[12] has been designed as a service that provides the scientific community with a Wiki and discussion forum for publications. *crowdLabs*[13] and *myExperiment*[14] have been proposed as environments for sharing workflows that describe computational experiments, data analyses and visualizations. However, in practice, there is a still long way to go for achieving effective and collaborative innovations in the research practices. In particular, surprisingly, the Computer Science research communities have not been successful, so far, to make the best use of or effectively exploit the availability of the recent advances in Web technologies to establish platforms and form driving forces that can address the above mentioned challenges and implement functional and widely-used collaborative experimental evaluation and benchmarking platforms that can dynamically evolve and exploit the power of the crowd.

In this article, we present the design and the implementation details of the *Liquid Benchmarking* platform [35] as a novel research infrastructure that provides cloud-based, collaborative and social environment that attempts to tackle the above mentioned challenges and obstacles by facilitating the *democratization*, *socialization* and improving the quality of the performance evaluation and benchmarking processes in the Computer Science research domain. In particular, we summarize the main contributions of our presented platform as follows:

- The platform can significantly reduce the effort and time for executing performance evaluation experiments by facilitating the process of sharing the experimental artifacts (e.g., software implementations, benchmarking tasks, computing environments) and supporting its end users to easily design, mashup and execute the experiments with zero installation or configuration efforts.
- The platform supports for searching, comparing, analyzing and visualizing 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 collaborative and social features of the platform enable 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.

In addition, we present the implementation details of four

[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/
[9] http://wp.sigmod.org/
[10] http://www.dbms2.com/
[11] http://semwebtec.wordpress.com/
[12] http://www.pubzone.org
[13] http://www.crowdlabs.org/
[14] http://www.myexperiment.org/home

benchmarking case studies that have been fully realized and made available via the *Liquid Benchmarking* platform. The remainder of this paper is organized as follows. Section II motivates the practicality and significance of our platform by illustrating sample scenarios. Section III discusses some of the fundamental obstacles for conducting trustable and conclusive experimental evaluation or benchmarking research in the Computer Science filed. In Section IV, we describe the main entities and the conceptual model of the *Liquid Benchmarking* platform. The architecture and the implementation details of the platform is presented in Section V. Four Liquid Benchmarking case studies are presented in Section VI before we conclude the paper in Section VII.

II. Motivating Scenarios

In this section, we present two sample scenarios that motivate the practical importance of our presented *Liquid Benchmarking* platform as follows.

Scenario 1: $Alan$ is a graduate student in one of the world reputable research groups on data management systems. He and his advisor are researching on developing novel efficient techniques for querying graph-based biological databases. $Alan$ has been recommended by his advisor to perform a survey on the related literature and conduct an experimental evaluation for assessing the performance characteristics of the state-of-the-art. During this activity, $Alan$ got overwhelmed with a large number of literature which are reporting scientific proposals for techniques and approaches to tackle the problem of interest. As a result of an extensive research task, $Alan$ has been successful on getting the access to the software implementation for some of proposed approaches and techniques in the literatures while he exploited his technical software development skills to re-implement a set of the important approaches which were reported in the literature, according to their reported description, but that have no available software implementations. After a year of effort, $Alan$ prepared all the requirements to conduct a benchmarking study which assesses and compares between some of the proposed techniques for tackling his problem of interest. This experimental evaluation study supported Alan to gain useful insights for achieving his primary objective. Apparently, this is very time and effort consuming task (in addition to some other various obstacles which will be discussed in more details in Section III). In practice, the accuracy and the quality of the outcomes of such conducted experimental evaluation activity has been constrained with the amount of effort, time and attention which has been dedicated by two researchers: $Alan$ and his advisor, through this research activity. In general, it is common that graduate students in the various research areas of the Computer Science field usually go through a similar process at the initial stages of their research work. Unfortunately, prospect graduate students in the same domain, across the world, may not be able to leverage, extend or improve *Alan*'s effort unless there is an effective and workable solution or platform that allow $Alan$ to share the artifacts of his study and enable other students and researchers to collaborate on following up and contributing to this sort of experimental evaluation research. In addition, after one year, $Alan$ may not be able to reproduce his own results or explain them. In practice, constant time pressure and strict submission deadlines usually push the scholars to favor timely results over spending enough time on documenting the experiments and data traceability.

Scenario 2: $John$ is one of the active researchers in the domain of graph databases. During his research, $John$ got interested in benchmarking the state-of-the-art of the indexing and querying approaches for graph databases. As a result, *John* allocated about 24 weeks of his time in the following activities:

- Establishing a large corpus of graph databases that have various characteristics.
- Searching for the available software implementations of graph indexing and querying techniques in addition to implementing some of the techniques that have been presented in the literature but they have no available software implementation.
- Conducting extensive experiments to assess the performance characteristics of various proposed techniques, which were proposed in the literature, using the established collection of graph datasets and analyze their results.
- Documenting the results of the conducted experiments, sharing the artifacts of the study via a public web page and writing up a journal publication that disseminate the results and lessons of his benchmarking study.

Following its release, *John*'s benchmarking study has attracted a lot of interest from the research community of graph databases where some of the active researchers in this domain had communications with $John$ to inquiry about various aspects of the experimental study or seeking some advices in reproducing the results of some of the reported experiments. However, unfortunately, these communications remained offline in *John*'s mail box. After sometime, John has moved to a new position and his research interest shifted to another research area. Hence, he become less responsive to inquiries from researchers in the graph indexing and querying domain about his benchmarking study. In addition, the results of his benchmarking study has become out-of-date after the introduction of novel approaches and techniques that tackle the same problem and the improvement of formerly investigated approaches by John. In practice, effectively and cumulatively exploiting *John*'s effort calls for joint efforts from other active scholar in the graph indexing and querying domain in addition to an the availability of an adequate platform that can facilitate and support such efforts.

III. Benchmarking Challenges in Computer Science

In comparison to more traditional disciplines (e.g., natural sciences), computer science is considered a much younger discipline which is usuall said to have a somewhat sloppy relationship with the repeatability of published results [13]. In practice, each computer science scholar could share a story about of a failed attempt to reproduce the results of a some top-notch paper [13]. For example, Collberg et al [11] have reported that they have failed, in many cases failed, to systematically replicate artifacts from highly ranked research papers. In this section, we discuss some of the remarkable obstacles for conducting trustable and conclusive benchmarking studies in the Computer Science research field as follows.

- *Limited reproducibility of reported experimental results*: In an ideal world of Computer Science research, the authors of a research outlet document the details of their contributions in the manuscript and publicly provide the binaries/source codes of their software implementation with the other related software atrifacts (e.g., experimental datasets) to the other researchers so that they can be exploited for reproducing the reported results in their publication. This ideal process would provide several advantages. For example, other researchers in the same domain of the study would be able to *independently* asses the performance characteristics of the provided software implementation using other experimental setups (e.g., datasets, computing resources) in order to verify the reported claims and make sure that there is no hidden aspects which can affect the accuracy of the reported experimental results. In addition, other researchers can exploit this available software artifacts as a valuable starting point to evaluate and assess the significance of their own proposed contribution. One of the interesting examples for the value of such independent evaluation studies is the study of Sidirourgos et al. [39] where they have reported about an independent assessment of the published result by Abadi et al. in [4] which described an approach for implementing a vertically partitioned DBMS for Semantic Web data management. The outcomes of this independent assessment revealed many interesting aspects. For instance, in [4] Abadi et al. reported that the performance of binary tables is superior to that of the clustered property table for processing RDF queries while Sidirourgos et al. [39] reported that even in column-store database, the performance of binary tables is not always better than clustered property table and depends on the characteristics of the data set. In addition, the experiments of [4] reported that storing RDF data in column-store database is better than that of row-store database while [39] experiments have shown that the gain of performance in column-store database depends on the number of predicates in a data set. A main lesson from this example is that we cannot really be sure that published research results are accurate and comprehensive even if they were reported by the best scientists and went through the most rigorous peer review process. However, can have more confidence on these results if others can repeat the same experiments and obtain similar results [16]. However, it should be noted that such repetitions are considered part of the scientific process and they do not represent any mistrust for the scholars who published the original results. Instead, they represent part of the scientific process which aims of gaining more confidence in the original results or to provide more insights that can specify or delimit the range of their applicability.
  In practice, unfortunately, the research world is not usually following this ideal process. For instance, Sakr [32] has performed a benchmarking study for the state-of-the-art of XML compression techniques [1]. The results of this study have shown that many XML compression techniques which were presented in the literature have no available software implementations and thus it is hard or not straightforward to assess their performance characteristics. Collberg et al. [11] have also reported in their study for highly ranked research papers that when software was available, with a percentage of only 44% of the cases, it was difficult to have it running. Clearly, such limitation prevents the researchers from confirming the reproducibility of the reported figures in the original publications and hinders the chances of conducting comprehensive comparisons among the whole set of the proposed techniques for tackling the same research challenges. Recently, some groups have organized initiatives to establish open challenges in various research domains (e.g. Semantic Web Service Challenge, Semantic Web Challenge, Information Retrieval). In addition, recent editions of SIGMOD conference started to offer the opportunity for the researchers of the published manuscripts to evaluate their software using the experimental datasets to reproduce the reported experimental results. Unfortunately, so far, the repeatability reports of the SIGMOD conference have shown limited success on achieving this goal due to several reasons [7], [26], [27].

- *The dynamics and continuous evolution of the state-of-the-art*: In practice, conducting an independent, comprehensive and conclusive benchmarking study for the-state-of-the-art in any research area is a very useful but also a challenging task which involves considerable time, effort and resources. For example, it may require designing different scenarios, choosing different datasets and evaluating the performance characteristics using various metrics. Therefore, some journals (e.g., the *Elsevier Performance Evaluation* Journal[15]) focus their scope of interest around manuscripts that consider this type of experimental evaluation research. In 2008, the reputable VLDB scientific venue initiated a new experimental analysis research track that focuses on analyzing the advantages and drawbacks of various techniques which are tackling the same research challenge. Although this type of research publications are useful, they suffer from a main limitation that they reflect *snapshots* for the state-of-the-art at the time of their conduct. However, by default, the research contributions in any research area are always *evolving* and *dynamic*. For instance, novel techniques which are designed to address the same research challenge of a formerly published snapshot publication can be proposed or the performance characteristics of formerly evaluated techniques can develop and improve. Hence, these publications may go out-of-date after a relatively short period of their release. Assuming that the results of such benchmarking studies can be maintained on web pages, *continuous* evolving and maintenance of the reported results may require too much effort from the authors who may loose interest in re-executing the same task after sometime. Finally, it is not practically recommended in the current very dynamic environment to spend several years in conducting a set of benchmarking experiments in a certain research domain. In particular, nowadays, the development of such benchmarking studies should be fast, dynamic and reactive in order to be valuable.
- *Constraints on the availability of computing resources*:

[15]http://www.elsevier.com/locate/peva

In various domains, performing conclusive benchmarking study may require huge computing resources. In addition, conducting experimental evaluations may require experimenting with various configurations for the computing environments in order to reflect the various configurations of computing environments in real-world scenarios. In practice, the availability of such computing resources requirements for researchers who are aiming to conduct a benchmarking study in their home labs/environments can be limited which consequently can limit or prevent their capacity to conduct comprehensive and insightful benchmarking studies. For instance, Pavlo et al. [30] described a benchmarking study that compares between the performance and the development complexity of parallel databases and MapReduce in executing *large-scale* data analysis jobs. In practice, reproducing the results of the experiments which has been reported in this publication by other researchers is a very challenging task due to the high and demanding configurations of the the testing computing environment. In particular, the original experiments which have been reported in this publication were conducted using a computing cluster of about hundred machines. In general, conducting a *fair* and *apples-to-apples* comparison among any alternative software implementations would require executing the experiments using *exactly* similar computing environments and the same experimental artifacts. In addition, it is crucial that an experimental evaluation study test the performance characteristics of hardware components and subsystems in a realistic and meaningful way. Therefore, ideally, researchers should have the facility to access shared computing resources where they can compare/evaluate the various software, under study, consistently. The adequate configurations of such experimental computing environments should be also decided *collaboratively*.

- *Not enough standard benchmarks are available or widely-used*: A benchmark is a *standard* test or set of tests which is utilized to compare/evaluate different techniques that have a shared objective to address a certain research challenge. In practice, the unavailability of a standard benchmark in a specific research issue represents a major source of hardship for the researchers who want to comprare/evaluate their contribution in this domain and consequently leads to reporting about various adhoc experimental results in the various publication which documented research efforts that attempted to tackle this research challenge. In principle, a benchmark usually consists of a motivating scenario, a set of benchmarking tasks in addition to specifying a set of performance evaluation metrics. In principle, limited number of benchmarks usually succeed on gaining wide acceptance and achieving good success in their target research community. For instance, in the database research community, some benchmarks were successful on achieving such success including:
  - The TPC group of benchmarks for evaluating the performance characteristics of transaction processing in relational database management systems [3].
  - The oo7 benchmark [8] which has been presented as a standard benchmark for evaluating the performance characteristics of object-oriented database systems.
  - The XML Benchmark Project (XMark) [38] which has been used as a mean to evaluate the performance characteristics of XML data management systems.

  However, on the other hand, there are still many other research aspects in the database research community which are in a significant need for defining standard benchmarks that fulfil the requirements of the researchers in assessing the impact of their contributions (e.g. graph databases, RDF databases, big data processing systems, NoSQL databases, scientific databases) [33], [34], [40]. In practice, for any benchmark to be successful, it needs to gain wide acceptance by its target community. Hence, the motivating scenario of the defined benchmark should be *simple*, the set of testing tasks and performance metrics should be *complete* and *generic* [12]. In addition, such standard benchmarks should satisfy other general and important qualities such as *portability*, *relevance*, *scalability* and *extensibility* [22]. In practice, it is challenging that a single benchmark can reflect the various usage scenarios and achieve all these quality goals. Therefore, it is common that many research domains require defining *microbenchmakrs* [5] that have deep focus in a specified detailed aspects. In principle, a well-designed benchmark in a certain domain is usually very useful to the active researchers in that domain as it constitutes the fundamental basis for evaluation and comparing their research contributions. Therefore, they become able to specify the advantages and disadvantages of their contribution which can effectively inspire their plans for the various directions of improvement. However, designing a successful benchmark is a quite challenging task which is usually not easily achievable by a single researcher or research group. Ideally, effectively tackling the challenge of establishing standard and successful benchmarks would require collaborative efforts from various groups of peer researchers within the target domain of the benchmark.

IV. Conceptual Model

The primary objective of the Liquid Benchmarking platform is to provide a cloud-based and social platform which can simplify and democratize the job of computer science scientific scholars in conducting solid experimental evaluations with high quality. In particular, the features of this platform is designed to provide scientific scholars with various mutual services including:

- Establishing repositories of related and competing software implementations where these implementations can be executed as software services that involve no installation or configuration requirements at the users side.
- Sharing testing computing environments.
- Collaboratively defining, discussing and evolving the specifications of standard benchmarks to assess the competing software implementations.
- Providing the end-users with an environment that supports easily creating and executing testing experiments and share their results.

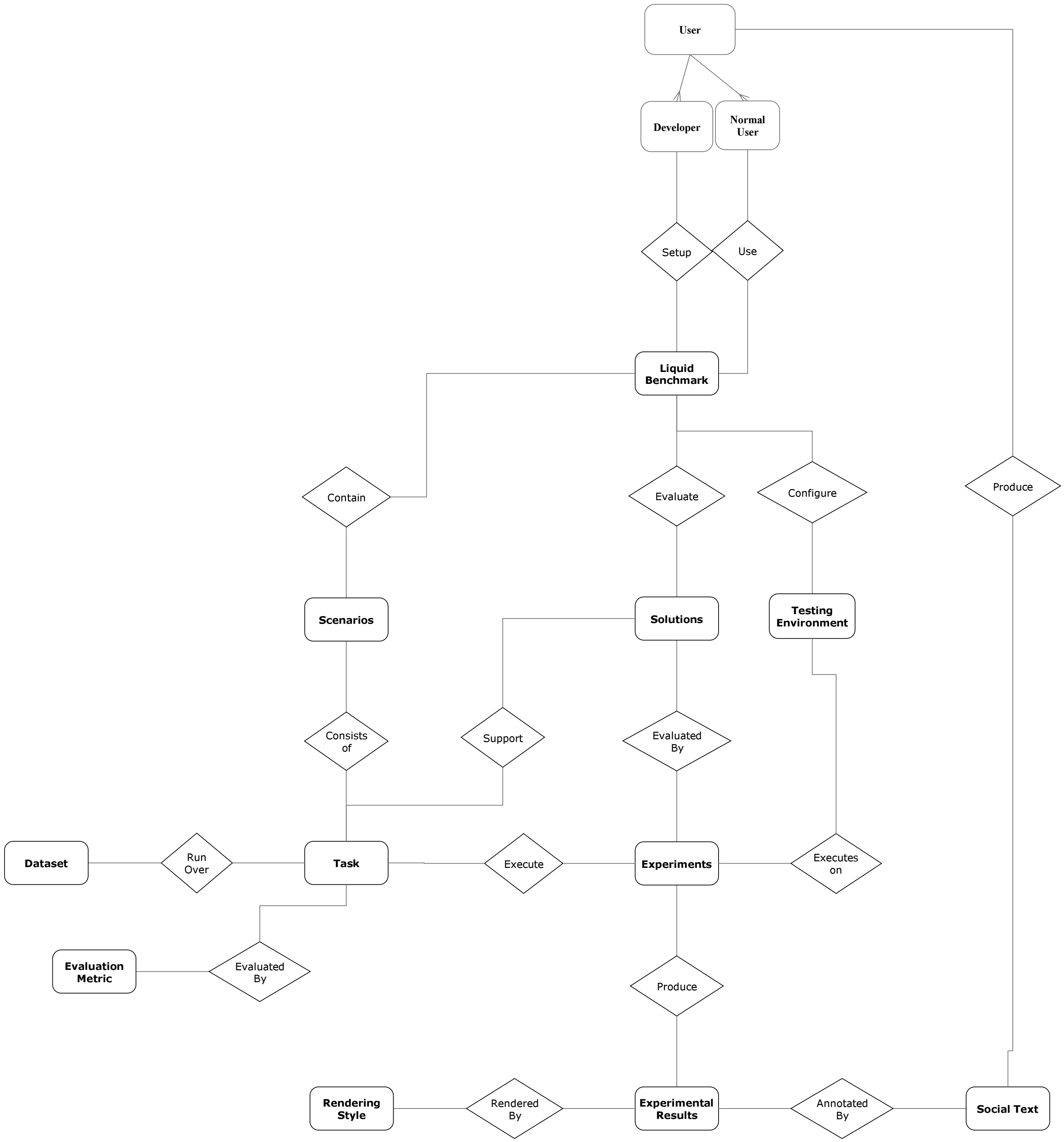

Fig. 1: Conceptual Model of Liquid Benchmarks

Figure 1 illustrates an overview of the conceptual model for the main entities of the Liquid Benchmarks. In this model, we differentiate among two types of users: *developer user* (benchmark developing committee) and *normal user*. Developer users represent the set of researchers who have the privilege to participate in the collaborative environment for defining the configurations of the different components of the benchmark (e.g. datasets, tasks, evaluated software) while normal users are only allowed to use the defined configurations of the benchmark to run their test experiments. However, normal users can be optionally allowed to do some configuration tasks such as: uploading their own datasets or defining their own tasks for running specially defined experiment in a *private* area which is separated from the public setup of the benchmarks. In particular, each liquid benchmark is configured by defining the following main components:

- **Scenarios**: In principle, each liquid benchmark consists of at least one scenario which models a use case that focuses on evaluating some aspects of the competing softwares in the target domain (e.g. MacroBenchmark or MicroBenchmark). In particular, each scenario is described by a *Service Schema* that defines the set of parameters (inputs and outputs) which need to be defined for interfacing with the services of the evaluated softwares.
- **Evaluated Solutions**: The set of competing software implementations (e.g. techniques, algorithms, systems) which are developed to tackle the specific problem of the liquid benchmark. In practice, each software implementation may have various *versions*. Each of these versions is treated as a separate (but linked) competing solution. Each solution need to register the set of its supported tasks in order to avoid the running of many failing tasks.
- **Task(s)**: Describes a set of operations which should be executed by the competing software implementations (e.g. update operations, queries, compressing operations). In practice, each operation usually assesses one or more target evaluation aspects which is in the scope of the benchmark specifications.
- **Metric(s)**: Represents the measures of evaluating the performance characteristics of the competing software implementations in performing the various defined tasks of the benchmark (e.g. execution time, response time, throughput). In particular, metrics represent the basis of comparing the competing software implementations.
- **Testing Environment(s)**: Represents a set of different configurations for computing environments (e.g. operating system, CPU, disk space, main memory) that reflect various real-world scenarios.

V. Platform Architecture and Implementation

In principle, the features and design decisions of the implementation of the Liquid Benchmarking platform[16] combine the facilities provided by different emerging Web technologies which are described as follows:

- *Software-as-a-Service*: The platform uses the RESTful architectural style as an effective software distribution technique [42] in which software implementations can be installed on the hosting computing environment and made available via an application programming interface to the end-users via the Internet. This technology 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 [29].
- *Cloud Computing*:
Benchmarking in practical computer science requires more than just data and code, however, it also requires an appropriate and *shared* or *identical* computing environment in which to run experiments. The platform exploits cloud computing as an emerging effective technology for broad sharing of hardware resources and computing environments over the Internet [15]. In particular, virtualization is a key technology of the cloud computing paradigm which 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 [36], [6]. 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 [31]. 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 [31]. 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 and social Web capabilities (e.g. tagging, forums, user comments) that support human interaction and facilitates the establishment 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 manner [14]. Most important, the platform supports sharing the performance evaluation and benchmarking artifacts (e.g., software implementations, datasets, virtual machines) in a *workable* environment.

Figure 2 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 she/he can *mash up* the components (e.g., services, computing environments, tasks, metrics) of the experiment in a *drag and drop* style (Figure 3). In principle, according to the configuration of the components of the liquid benchmark, end users can design and run their *experiments* where each experiment is specified by: the *solution(s)* (software implementation(s)) to be evaluated, the *task* to be executed

[16]The implementation of the Liquid Becnhmarking platform is available onhttp://liquidbenchmark.net:8080/Liquid/

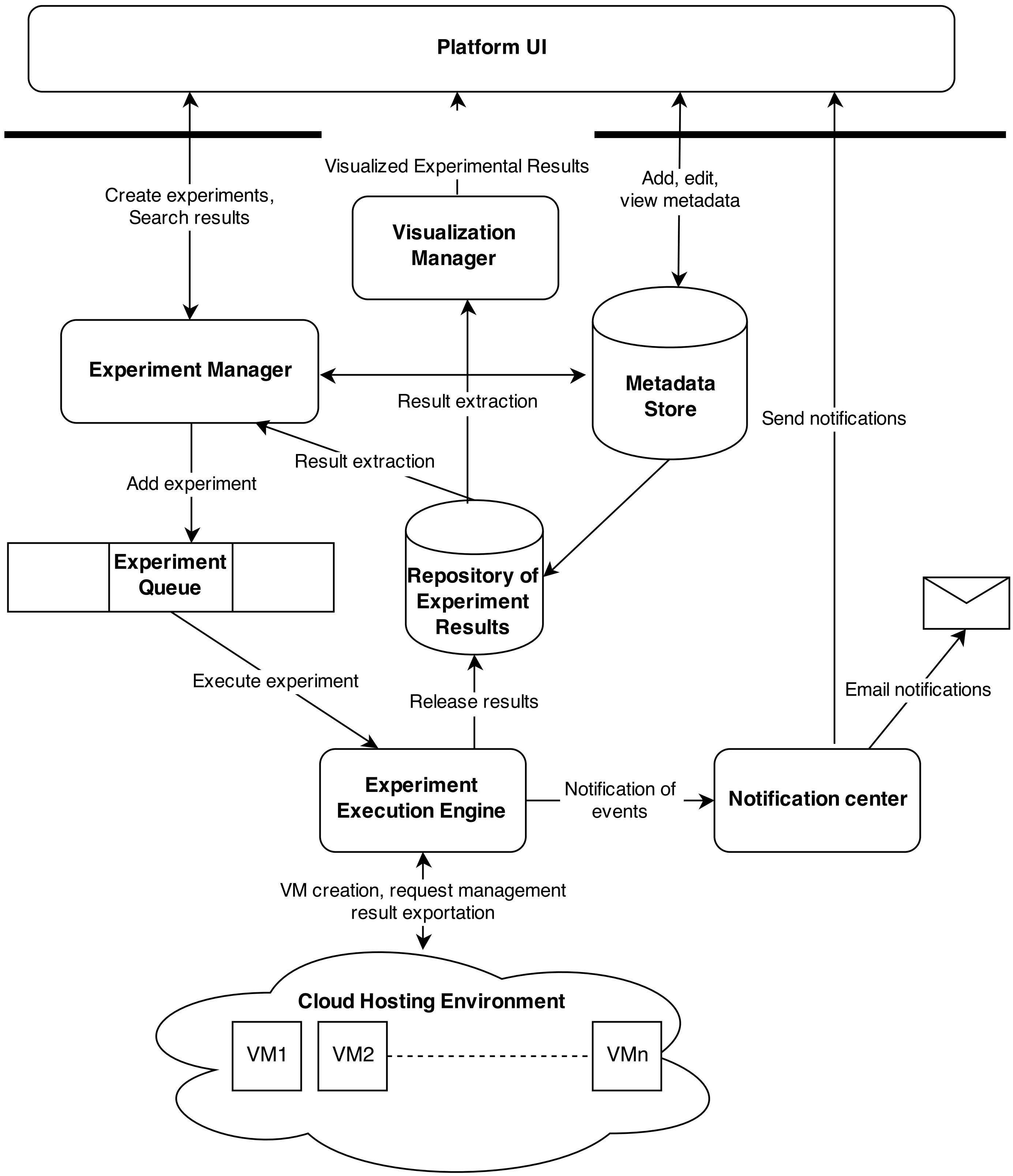

Fig. 2: Platfrom Architecture

with the associated instantiation of the parameters of the service schema, the selected metrics for evaluation and the testing *environment* which will be used for running the experiment. The platform user interface also provides

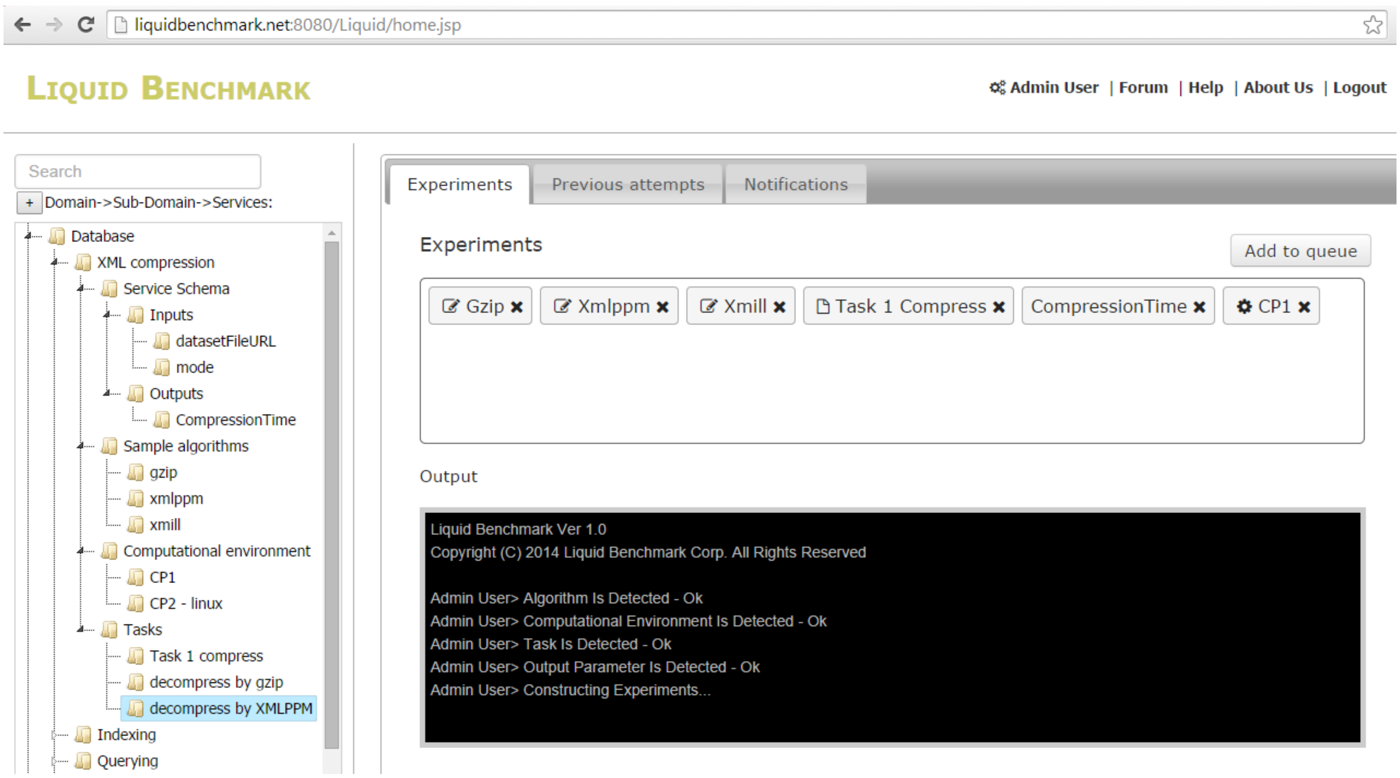

Fig. 3: Screenshot: Mashing Up an Experiment

the end-users with other facilities including 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 various components of the benchmark (e.g., services, service schema, tasks, virtual machines).
- **Experiment Manager**: The experiment manager receives the specification of the user-defined experiment, which is configured by the Liquid Benchmark user interface, an registers this experiment 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.
- **Repository of Experiment Results**: This is a central repository that 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 formerly 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. line charts, column charts) for presenting and comparing the results (metrics) of the selected experiments by the end-users (Figure 4).

VI. Case Studies

In this section, we present four benchmarking case studies which have been realized using the Liquid Benchmarking platform[17] on the following domains:

- *XML compression*[18]: This case study is based on the benchmark of XML compressors (e.g., XMill [24], Gzip, Bzip, XMLPPM [9]) that has been presented in [32]. In particular, this case study provides services for the implementation of nine XML compression tools with benchmarking tasks over a large XML corpus that covers the different types and scales of XML documents. This case study evaluates the XML compressors by three

[17] The full documentation for using the platform is available on http://wiki.liquidbenchmark.net/

[18] The full documentation and screencast of this case study is available on http://wiki.liquidbenchmark.net/doku.php/casestudy-xmlcompression

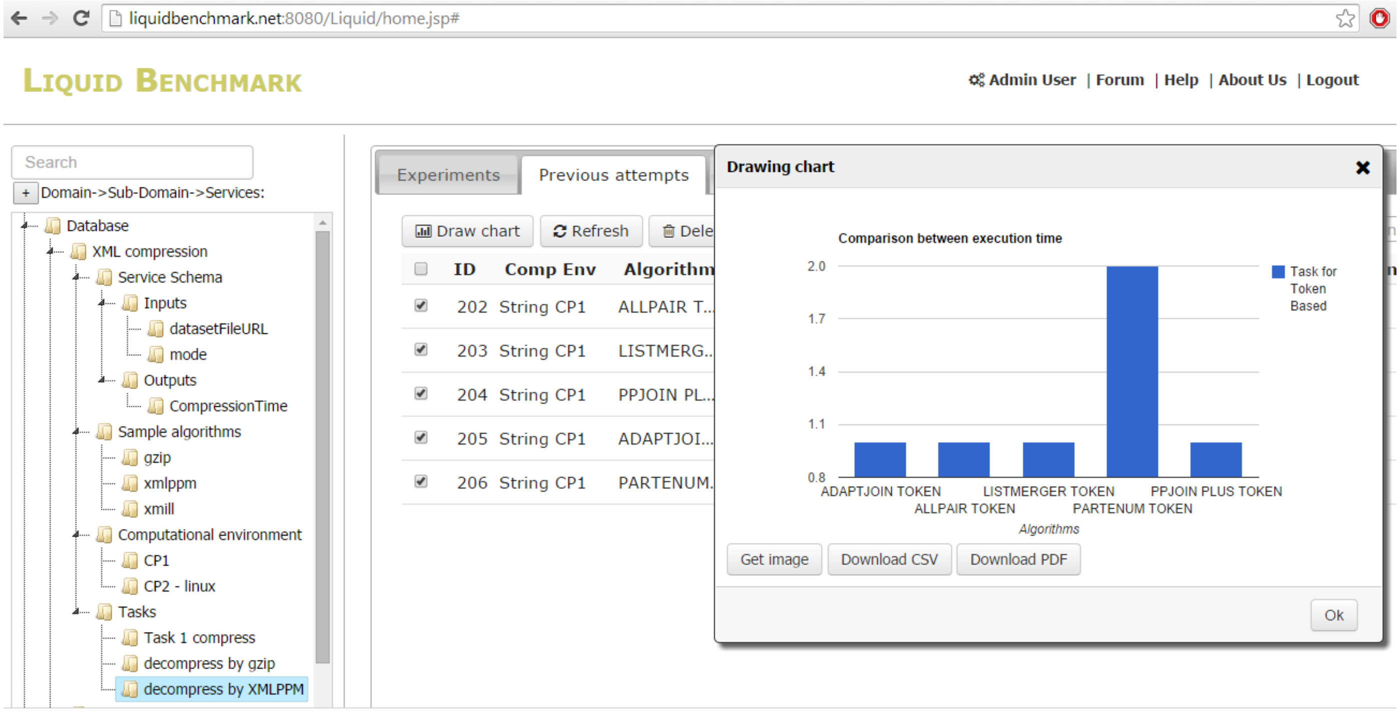

Fig. 4: Screenshot: Comparing and Visualizing Experimental Results

different metrics: compression time, decompression time and compression ratio.

- *Graph indexing and querying*[19]: This case study implements the *iGraph* framework [19], [20] for evaluating various graph indexing and querying techniques (e.g. Closure-Tree [21], gIndex [43], TreePi [45]). 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*[20]: An implementation for the recent evaluation and comparison study which is presented by Jiang et al. [23]. 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.
- *Reverse K Nearest Neighbors (RkNN)*[21]: An implementation for the recent evaluation and comparison study which is presented by Yang et al. [44]. The case study provides the implementation of various Reverse k Nearest Neighbors Query Processing algorithms over various experimental datasets.

Each of our case studies is deployed in two cloud environments: the Amazon public cloud environment[22] with its various cloud services (e.g., Simple Storage Service (S3[23]), Elastic Compute Cloud (EC2[24])) in addition to our own private cloud environment which is managed by the *OpenStack* platform[25]. However, the platform can be easily adopted to run over other cloud environments (e.g., *CloudStack*[26], *Eucalyptus*[27]). In addition, each case study is configured using two different testing environments (virtual machines): The first environment is configured with high computing resources while the other environment is configured with limited computing resources in order to imitate the various real world scenarios. Furthermore, authenticated users of our platforms can access various services of the platform (e.g., searching the repository of results, creating and running experiments) via our provided RESTful interfaces and API-based SDK[28]. Tho social features of our platform has been implemented the open source social network platform, *elgg*[29].

VII. Conclusion

In principle, the field of practical computer science is suffering from the repeatability problem of the research results [11] which represents a keystone of the scientific process. It is common that the results of experiments tend to reside in some folders or repositories which have never been documented thoroughly. Several reasons are behind this problem but time pressure is the most prominent of them. In practice,

[19]The full documentation and screencast of this case study is available on http://wiki.liquidbenchmark.net/doku.php/casestudy-graph-indexing-querying
[20]The full documentation and screencast of this case study is available on http://wiki.liquidbenchmark.net/doku.php/casestudy-string-similarity-join
[21]The full documentation and screencast of this case study is available on http://wiki.liquidbenchmark.net/doku.php/reverse-k-nearest-neighbors
[22]http://aws.amazon.com/
[23]https://s3.amazonaws.com/
[24]http://aws.amazon.com/ec2/
[25]http://www.openstack.org/
[26]http://cloudstack.apache.org/
[27]https://www.eucalyptus.com/
[28]http://wiki.liquidbenchmark.net/doku.php/RESTful-interface
[29]http://elgg.org/

documentation of experiments and data traceability needs valuable work time, while publish or perish and strict conference deadlines call for timely results. In practice, the Web has dramatically enhanced the people's ability to share knowledge, ideas and contributions. We believe that the Computer Science research community should have the leadership in having such *scientific* collaborative environments that can significantly develop and improve the capacity of the scientific communities on deeply understanding the details of their research challenges, have careful, clean and insightful analysis for the state-of-the-art that can support them for developing new effective approaches, techniques and solutions.

In this article, we presented the design and implementation details of the *Liquid Benchmarking* platform that relies on the current advances in the Web technologies to provide *collaborative* Web-based platforms that democratize and socialize the key tasks of evaluating, comparing and analyzing the *continuous* scientific contributions in different domains of the Computer Science field. We believe that our platform can effectively exploit the increasing human power which are participating in the Computer Science research efforts and distributed over the world. In particular, we argue that our platform can empower the Computer Science research communities with many capabilities such as:

- Developing *focused* and centralized repositories for related software implementations [2] and their experimental results. These repositories can serve as a very positive step towards tackling the experimental *reproducibility* challenge in the Computer Science field.
- Facilitating the establishment of shared computing resources environments that can be exploited by different active contributors in the same domain who reside in different parts of the world.
- Providing *workable* environments to collaboratively establish standard benchmarks that can be widely utilized for achieving insightful evaluation for alternative research efforts. These environments can help researchers to optimize their time in assessing and improving the quality of their contribution. Having such environments will discourage authors from publishing paper with adhoc or poor experimental results.
- Facilitating *collaborative* maintenance of experimental studies to guarantee their *freshness*. This task can follow the same model of collaborative organization of international conferences or journals where each participating researchers or research groups in a specific community can play a volunteering managerial role for a specific period.
- Exploiting the *wisdom of the crowd* in providing feedbacks over the experimental results in a way that can provide useful insights for tackling further problems and improving the state-of-the-art.
- Creating a *transparent* platform for scientific *crediting* process based on collaborative community work.
- Establishing concrete foundations and feasible environments for providing *provenance* services [28] for scientific experimental results and time-analysis services for the evolution of research efforts.

Therefore, we hope that our platform can serve as the foundation for a fundamental rethinking of the experimental evaluation process in the Computer Science field. As a future work, we are planning to implement more case studies using our platform and make them available for the research community. In addition, we are planning to add more features of the social aspect of the platform with careful consideration to important details such as credit attribution and data anonymization [17].

ACKNOWLEDGEMENT

This project was funded by the National Plan for Science, Technology and Innovation (MAARIFAH) – King Abdulaziz City for Science and Technology - the Kingdom of Saudi Arabia – award number (11-INF1992-03). The authors also, acknowledge with thanks Science and Technology Unit, King Abdulaziz University for technical support.

REFERENCES

[1] Benchmark of XML compression tools. http://xmlcompbench.sourceforge.net/.

[2] SourceForge: A free repository of open source software. http://sourceforge.net/.

[3] Transaction Processing Performance Council. http://www.tpc.org/default.asp.

[4] Daniel J. Abadi, Adam Marcus, Samuel Madden, and Katherine J. Hollenbach. Scalable Semantic Web Data Management Using Vertical Partitioning. In *VLDB*, pages 411–422, 2007.

[5] Denilson Barbosa, Ioana Manolescu, and Jeffrey Xu Yu. Microbenchmark. In *Encyclopedia of Database Systems*, page 1737. 2009.

[6] David Bermbach, Liang Zhao, and Sherif Sakr. Towards Comprehensive Measurement of Consistency Guarantees for Cloud-Hosted Data Storage Services. In *TPCTC*, 2013.

[7] Philippe Bonnet, Stefan Manegold, Matias Bjørling, Wei Cao, Javier Gonzalez, Joel A. Granados, Nancy Hall, Stratos Idreos, Milena Ivanova, Ryan Johnson, David Koop, Tim Kraska, René Müller, Dan Olteanu, Paolo Papotti, Christine Reilly, Dimitris Tsirogiannis, Cong Yu, Juliana Freire, and Dennis Shasha. Repeatability and workability evaluation of SIGMOD 2011. *SIGMOD Record*, 40(2):45–48, 2011.

[8] Michael J. Carey, David J. DeWitt, and Jeffrey F. Naughton. The oo7 Benchmark. In *SIGMOD*, pages 12–21, 1993.

[9] James Cheney. Compressing XML with Multiplexed Hierarchical PPM Models. In *Proceedings of the Data Compression Conference (DCC)*, page 163, Washington, DC, USA, 2001. IEEE Computer Society.

[10] Fernando Seabra Chirigati, Matthias Troyer, Dennis Shasha, and Juliana Freire. A computational reproducibility benchmark. *IEEE Data Eng. Bull.*, 36(4):54–59, 2013.

[11] Christian Collberg, Todd Proebsting, Gina Moraila, Zuoming Shi, and Alex M Warren. Measuring Reproducibility in Computer Systems Research. Technical report, 2014.

[12] Alain Crolotte. Issues in Benchmark Metric Selection. In *TPCTC*, pages 146–152, 2009.

[13] Christian Dietrich and Daniel Lohmann. The dataref versuchung: Saving Time through Better Internal Repeatability. *Operating Systems Review*, 49(1):51–60, 2015.

[14] Peter Dolog, Markus Krötzsch, Sebastian Schaffert, and Denny Vrandecic. Social Web and Knowledge Management. In *Weaving Services and People on the World Wide Web*, pages 217–227, 2008.

[15] Thomas Erl, Ricardo Puttini, and Zaigham Mahmood. *Cloud Computing: Concepts, Technology & Architecture*. Prentice Hall, 2013.

[16] Dror G. Feitelson. From repeatability to reproducibility and corroboration. *Operating Systems Review*, 49(1):3–11, 2015.

[17] Juliana Freire, Philippe Bonnet, and Dennis Shasha. Exploring the Coming Repositories of Reproducible Experiments: Challenges and Opportunities. *PVLDB*, 4(12):1494–1497, 2011.

[18] Juliana Freire, Philippe Bonnet, and Dennis Shasha. Computational re-

producibility: state-of-the-art, challenges, and database research opportunities. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD*, pages 593–596, 2012.

[19] Wook-Shin Han, Jinsoo Lee, Minh-Duc Pham, and Jeffrey Xu Yu. igraph: A framework for comparisons of disk-based graph indexing techniques. *PVLDB*, 3(1):449–459, 2010.

[20] Wook-Shin Han, Minh-Duc Pham, Jinsoo Lee, Romans Kasperovics, and Jeffrey Xu Yu. iGraph in action: performance analysis of disk-based graph indexing techniques. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1241–1242, 2011.

[21] Huahai He and Ambuj K. Singh. Closure-Tree: An Index Structure for Graph Queries. In *ICDE*, page 38, 2006.

[22] Karl Huppler. The Art of Building a Good Benchmark. In *TPCTC*, pages 18–30, 2009.

[23] Yu Jiang, Guoliang Li, Jianhua Feng, and Wen-Syan Li. String Similarity Joins: An Experimental Evaluation. *PVLDB*, 7(8):625–636, 2014.

[24] Hartmut Liefke and Dan Suciu. XMill: An efficient compressor for XML data. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 153–164. ACM, 2000.

[25] Stefan Manegold and Ioana Manolescu. Performance evaluation in database research: principles and experience. In *Proceedings of the 12th International Conference on Extending Database Technology (EDBT)*, page 1156, 2009.

[26] Stefan Manegold, Ioana Manolescu, Loredana Afanasiev, Jianlin Feng, Gang Gou, Marios Hadjieleftheriou, Stavros Harizopoulos, Panos Kalnis, Konstantinos Karanasos, Dominique Laurent, Mihai Lupu, Nicola Onose, Christopher Ré, Virginie Sans, Pierre Senellart, T. Wu, and Dennis Shasha. Repeatability & workability evaluation of SIGMOD 2009. *SIGMOD Record*, 38(3):40–43, 2009.

[27] Ioana Manolescu, Loredana Afanasiev, Andrei Arion, Jens Dittrich, Stefan Manegold, Neoklis Polyzotis, Karl Schnaitter, Pierre Senellart, Spyros Zoupanos, and Dennis Shasha. The repeatability experiment of sigmod 2008. *SIGMOD Record*, 37(1):39–45, 2008.

[28] Simon Miles, Paul T. Groth, Ewa Deelman, Karan Vahi, Gaurang Mehta, and Luc Moreau. Provenance: The Bridge Between Experiments and Data. *Computing in Science and Engineering*, 10(3):38–46, 2008.

[29] David Patterson and Armando Fox. *Engineering Software as a Service: An Agile Approach Using Cloud Computing*. Strawberry Canyon LLC, 2013.

[30] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, and Michael Stonebraker. A comparison of approaches to large-scale data analysis. In *SIGMOD*, pages 165–178, 2009.

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

[32] Sherif Sakr. XML compression techniques: A survey and comparison. *J. Comput. Syst. Sci.*, 75(5):303–322, 2009.

[33] Sherif Sakr. Cloud-hosted databases: technologies, challenges and opportunities. *Cluster Computing*, 17(2):487–502, 2014.

[34] Sherif Sakr, Anna Liu, and Ayman G. Fayoumi. The family of mapreduce and large-scale data processing systems. *ACM Comput. Surv.*, 46(1):11, 2013.

[35] Sherif Sakr, Amin Shafaat, Fuad Bajaber, Ahmed Barnawi, Omar Batarfi, and Abdulrahman H. Altalhi. Liquid Benchmarking: A Platform for Democratizing the Performance Evaluation Process. In *EDBT*, 2015.

[36] Sherif Sakr, Liang Zhao, Hiroshi Wada, and Anna Liu. CloudDB AutoAdmin: Towards a Truly Elastic Cloud-Based Data Store. In *ICWS*, 2011.

[37] J. D. Scargle. Publication bias: The File-Drawer problem in scientific inference. *Journal of Scientific Exploration*, 14(1):91–106, 2000.

[38] Albrecht Schmidt, Florian Waas, Martin L. Kersten, Michael J. Carey, Ioana Manolescu, and Ralph Busse. Xmark: A benchmark for XML data management. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB)*, pages 974–985, 2002.

[39] Lefteris Sidirourgos, Romulo Goncalves, Martin L. Kersten, Niels Nes, and Stefan Manegold. Column-store support for RDF data management: not all swans are white. *PVLDB*, 1(2):1553–1563, 2008.

[40] Michael Stonebraker. A New Direction for TPC? In *TPCTC*, pages 11–17, 2009.

[41] Young-Kyoon Suh, Richard T. Snodgrass, and Rui Zhang. Azdblab: A laboratory information system for large-scale empirical DBMS studies. *PVLDB*, 7(13):1641–1644, 2014.

[42] Erik Wilde and Cesare Pautasso, editors. *REST: From Research to Practice*. Springer, 2011.

[43] Xifeng Yan, Philip S. Yu, and Jiawei Han. Graph Indexing: A Frequent Structure-based Approach. In *SIGMOD*, pages 335–346, 2004.

[44] Shiyu Yang, Muhammad Aamir Cheema, Xuemin Lin, and Wei Wang. Reverse k Nearest Neighbors Query Processing: Experiments and Analysis. *PVLDB*, 8(5):605–616, 2015.

[45] Shijie Zhang, Meng Hu, and Jiong Yang. TreePi: A Novel Graph Indexing Method. In *ICDE*, pages 966–975, 2007.