A Collaborative Citizen Science Platform for Real-Time Volunteer Computing and Games

Poonam Yadav, *Member, IEEE*, Ioannis Charalampidis, Jeremy Cohen, John Darlington, and Francois Grey

***Abstract*— Volunteer computing (VC) or distributed computing projects are common in the citizen cyberscience (CCS) community and present extensive opportunities for scientists to make use of computing power donated by volunteers to undertake large-scale scientific computing tasks. VC is generally a noninteractive process for those contributing computing resources to a project, whereas volunteer thinking (VT) or distributed thinking allows volunteers to participate interactively in CCS projects to solve human computation tasks. In this paper, we describe the integration of three tools, the Virtual Atom Smasher (VAS) game developed by CERN, LiveQ, a job distribution middleware, and CitizenGrid, an online platform for hosting and providing computation to CCS projects. This integration demonstrates the combining of VC and VT to help address the scientific and educational goals of games like VAS. This paper introduces the three tools and provides details of the integration process along with further potential usage scenarios for the resulting platform.**

***Index Terms*— Citizen cyberscience (CCS), cloud computing, community grid, crowdsourcing, human computation, middleware, nonprofit sector, online games, parallel computing, real-time distributed computing, volunteer computing (VC), volunteer thinking (VT).**

I. Introduction

In the last few years, citizen cyberscience (CCS) has evolved as a new way of inspiring and supporting learning and participation in science. It provides a means for citizens, who may not have a scientific background, to interact with and contribute to scientific projects or studies. In many cases, such interactions are beneficial to both the scientist running the project and to the participating individuals who can gain new skills and knowledge in the process of supporting the project. Existing CCS projects are mainly categorized as volunteer computing (VC) or volunteer thinking (VT) projects, examples of this are provided in [1]–[3]. In a VT project, volunteers use their cognitive skill and knowledge to solve a part of a scientific problem. This type of project requires volunteers' active participation. In VC projects, volunteers contribute their computing resources to provide processing power to support one or more computationally intensive tasks within a project.

Projects generally have specific use cases that are considered to be ideally suited to either VC or VT. A project may have significant computational requirements that can best be served by farming out pieces of computation to the computers of volunteers, who have opted to take part in the project. This allows passive volunteering from the user perspective with the user's computer becoming available to undertake computation for the project when the user is not actively using their machine. Alternatively, a project may be structured to take advantage of the ability of humans to undertake tasks that are computationally difficult but very straightforward for a human to process. Examples might include identifying particular properties of some data by looking at a graph or spotting visual anomalies in an image. Such tasks can be quickly and accurately undertaken by humans, while reliably undertaking such tasks using code can be challenging and computationally intensive. These tasks are ideally suited to VT, where volunteers can actively participate in a project and assist the project owner in achieving their aims. Nonetheless, there are increasing numbers of use cases, particularly in the realm of scientific education, where VT tasks can be made more realistic and more educationally valuable if they make use of realistic or pseudorealistic data. For example, consider an online science teaching tool where real-world computation of scientific data that may be impractical to undertake on a single user's machine can be integrated into the project. The data generation is itself an ideal VC task and can be farmed out to the computers of one or more participants in the project to generate the required data to support a user interacting with the tool.

We therefore believe that integrating VT and VC into a single project can help to make interesting and engaging use cases for CCS projects. There are two main deployment scenarios for combining VC and VT within CCS projects. These have been presented in various CCS articles, an overview of which can be found in [2]–[4]. The first scenario is where VT tasks are independent of VC tasks or where the two different types of task do not require real-time interaction with each other. The second scenario consists of a real-time interaction between the VT and VC tasks. In the first scenario where VC and VT tasks do not need to interact with each other in real time or are completely independent, implementation is generally fairly straightforward and the two aspects of the project can be implemented separately, even though their implementations may ultimately be within

Manuscript received August 23, 2017; accepted September 28, 2017. Date of publication January 8, 2018; date of current version February 23, 2018. This work was supported by the European Commission for the Citizen Cyberlab project through the Seventh Framework Program under Grant 317705. *(Corresponding author: Poonam Yadav.)*

P. Yadav is with the Computer Laboratory, University of Cambridge, Cambridge CB3 0FD, U.K. (e-mail: poonam.yadav07@alumni.imperial.ac.uk).

I. Charalampidis is with CERN, 1211 Geneva, Switzerland (e-mail: ioannis.charalampidis@cern.ch).

J. Cohen and J. Darlington are with the Department of Computing, Imperial College London, London SW7 2AZ, U.K. (e-mail: jeremy.cohen@imperial.ac.uk; j.darlington@imperial.ac.uk).

F. Grey is with the Citizen Cyberlab, University of Geneva, 1205 Geneva, Switzerland (e-mail: francois.grey@unige.ch).

Digital Object Identifier 10.1109/TCSS.2017.2771479

the same codebase. The second scenario is more complex to implement since a common framework must be provided that can link the task lifecycles of both VT and VC tasks. By this we mean that when a VT task requires some data that must be generated to order, this must trigger a related VC task that must be scheduled such that results are provided to best support the interactivity requirements of the VT task the end user is undertaking. To the best of our knowledge, there is currently no such collaborative platform described in the existing literature and we refer readers to [2] and [5], which provide an overview of this paper.

In this paper, we detail the integration of three platforms, CitizenGrid [4], [6], LiveQ [7], [8], and the virtual atom smasher (VAS) [7], [9] game, to show an example of a CCS project, where VT and VC are integrated. In addition to this specific use case, we look at how the integration of two of these platforms, CitizenGrid and LiveQ, can provide a more general platform for developing CCS projects and applications that integrate VC and VT. In Section II, we discuss the motivations behind the integration of VC and VT. In Sections III and IV, we first present a brief overview of the three target platforms followed by details of the design for the resulting integrated system. Sections V and VI present implementation details and a set of usage scenarios for the integrated system. Section VII presents related work, and we describe our conclusions and details of future work in Section VIII.

II. Motivations

The design, implementation, and deployment of a citizen science project are driven by a goal, which is first defined by considering the project creator's motivation and intentions. The volunteers who participate in citizen science projects have different motivations and goals for their participation [10]. However, the factors that primarily determine their decision between participating in a VC or VT project depend on their scientific and educational interests and the time and computing resources that they have available. Integrating VT and VC tasks into one project and allowing volunteers to participate in either or both types of task could make the project more popular and attract more participation. At the same time, integrating both VT and computing into a project can significantly increase the complexity of building and running the project. Therefore, in this section, we present our three main motivational goals for the integration of VT and VC in a single CCS project.

A. Volunteer Engagement

Engaging and retaining volunteers in a project is a challenging task. In recent years, a number of research studies have been conducted to understand what motivates volunteers and what project factors influence their continued participation [10]–[13]. It is understood from various studies that gamification of VT tasks[1] keeps people more entertained and engaged in the project [10], [14], [15]. Therefore, incorporation of serious VT games with VC projects that generally have limited interaction will offer another approach to enhance user participation in VC projects and make them more rewarding to take part in.

[1]The use of gaming approaches to represent scientific human computation tasks.

B. Scientific Education

Gamification and VT task interfaces provide great learning and creativity opportunities for volunteers. CCS game players and volunteers learn and enhance their project specific scientific knowledge through hands-on experience with real tools. For example, in the VAS game, the players learn about particle physics by visualizing simulated results that are generated using equation parameters that the volunteers can alter. The simulations correspond to the real-world experiments being undertaken by physicists at CERN. The game players also learn about various new computing technologies, for example, how to work with the VirtualBox [50] virtualization software. Without the integration of VC, the computation that would be required to undertake the simulations may be too much for an individual user's computer system. This would either mean that the generation of results would take too long and spoils the game playing experience or that result generation using real-world parameters would not be possible at all.

C. Participation Diversity

Previously, VC projects have attracted only volunteers who wish to contribute spare computing resources to a project, but are not so interested to interact with the project directly. The VC platforms [6], [16] offer the ability for volunteers to gain straightforward access to a range of citizen science projects registered with them. This, combined with projects that integrate both VT and VC processes, opens up a range of opportunities to attract new volunteers who have time to play games and are also interested to participate in and gain understanding and knowledge of scientific processes and challenges. We categorize volunteer participation in two levels: single-mode participation (either computing or thinking only), or combined participation. With single-mode participation, while a project may integrate both VT and VC, it is not necessary for a volunteer to participate using both approaches to take part in the project. However, in combined participation, a volunteer has to use both approaches to take part in the project. In the VAS-CitizenGrid scenario, while the project integrates the aspects of VC and VT and both are required for the game to operate successfully, individual game players need not contribute their computing resources for the VC tasks, but they can still participate in the learning aspects of the project. Game players can take part in the project as part of a team where some individuals in the team may participate only as VC providers, others may participate from the VT perspective. It is also possible to be both a VT and VC participant—in this case, the user is considered to be providing combined participation. If game players are not contributing their own processing resources when playing the VAS game, their computing tasks can be processed by other team members' computing resources. A survey has shown

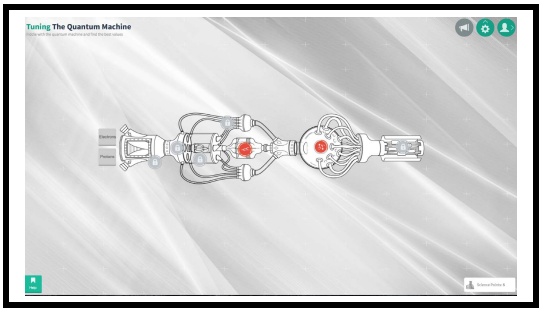

Fig. 1. VAS game "quantum machine" interface.

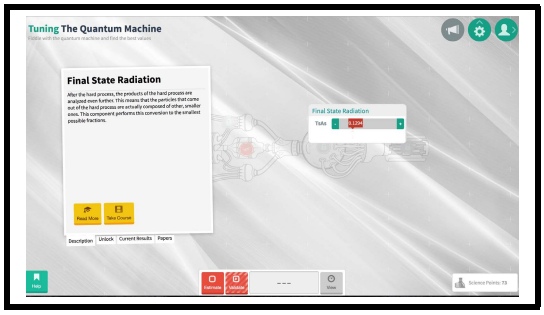

Fig. 2. VAS game interface showing the parameter setting and validation.

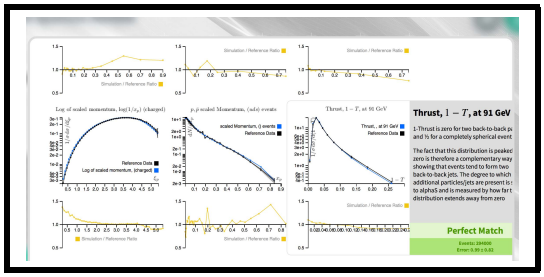

Fig. 3. VAS game interface showing simulation histograms.

that less than 10% of participants in VC projects are women, whereas, in online games, the participation of women and men is nearly equal [17], [18]. The survey results further support our strong motivation to integrate scientific gaming with VC projects to attract new participants to projects and ensure the involvement of a diverse range of volunteers.

III. Background

In this section, we provide a brief overview of each of the platforms used in this paper.

A. Virtual Atom Smasher

VAS [7], [9], [19] is a web-based physics game, developed at CERN, that aims to educate game players about particle physics through an interactive hands-on web-based environment. The game is based on the concept of a "Virtual Collider" that represents a real particle collider. The game front-end is developed as a web application. The interface contains interactive videos, rich pop-up explanations, and animations, which help players to understand the particle physics fundamentals, for example, high-energy particle collision. We use a set of screenshots from the web-based interface of the VAS application [19] in this section to give the reader an idea of the highly graphical nature of the game and how it presents scientific data and concepts to game players.

In Fig. 1, the game user interface shows a "quantum machine" with a number of locks, in principle it is an attempt to visualize the sequence of events that occur inside the event generation software which simulates a particle collision inside the Virtual Collider. The game player's goal is to unlock all the locks on the machine. In order to unlock parts of the machine, they need to spend science points that they have earned during the game. They earn science points by choosing and validating suitable tuning parameters (see Fig. 2).

The VAS game back-end is developed to tune Monto Carlo simulations of high-energy particle physics. The common problem in such cases is to find the correct values of a dozen parameters such that the simulation results match the observed results from the current and past particle colliders. Even though, for this classical minimization problem, there are already automated solutions (see [20]), we considered that the educational value of an interactive interface was much greater than using automated approaches.[2] The game front-end is developed as a web application using the latest HTML5 technologies for enhancing the educational experience and providing players with a step-by-step approach toward understanding the world of particle physics.

When players select and validate a parameter on the game interface, they request the execution of a Monte Carlo simulation to a set of distributed computer nodes. A player's task is to fine-tune the parameters of the CERN Virtual Collider simulator in such a way that the simulation presents the same results as those observed in the real experiments. The number of "science points" or "credits" a player gets depends on two factors. First, the values of the parameters they choose for the simulation. Generally, the first value chosen within a provided range is a random guess. However, in the subsequent steps, they can adopt a more guided approach, making decisions based on results provided by their previous chosen values. Fig. 3 shows a group of histograms that provide the results of previous simulations. The second factor for gaining credits is the speed at which simulations are processed. By processing their simulations more quickly, a user can try and validate more guesses at possible input values. If they provide a larger number of computing resources to run their simulations, they can undertake more simulations in a given period of time enabling them to gain more credits.

The VAS game provides many opportunities for those interested in learning about particle physics to gain some understanding of the field through a fun and competitive game-based environment. The integration of VC and VT in this environment offers an incentive for more volunteers to take

[2]This is one of the reasons why CERN has begun to grant access to the real scientific software and data that physicists use to anyone interested in the science of particle physics, via the portal http://opendata.cern.ch/.

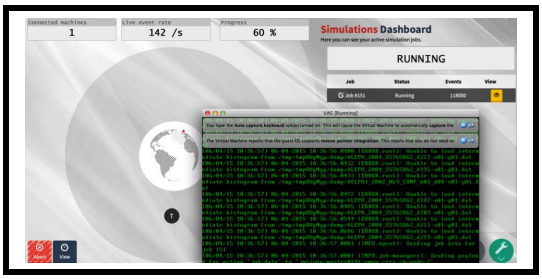

Fig. 4. VAS game interface showing real-time VC task progress status.

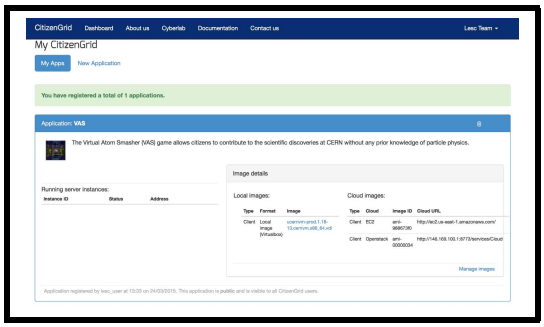

Fig. 5. CitizenGrid interface showing details of the VAS game as a registered VC project.

part in the game, and we believe that the approach used offers opportunities to provide a general educational environment for those interested in taking part in a variety of CCS games.

B. LiveQ

LiveQ is a job distribution and monitoring middleware with real-time interaction capabilities for VC projects. Generally, VC projects are based on a distributed client-server model [2] for VC task distribution and management. Instead of designing the distribution framework from scratch, many VC projects use generic middleware platforms that are specifically designed and implemented to support VC. For example, the SETI@Home project [21], a popular VC project, uses the BOINC [22], [23] middleware for its task distribution and management.

LiveQ offers real-time feedback and control of tasks and simplifies the process of managing large numbers of tasks and the integration of their results. LiveQ maintains a record of all the VC resources connected to it any time, which makes it a suitable choice for a game-based integrated project. The VAS game uses LiveQ for its task management.

Fig. 4 shows the VAS game interface displaying real-time task progress status. It also shows the number and details of VC client machines that are undertaking processing of tasks on behalf of the game player currently logged in to the interface.

C. CitizenGrid

CitizenGrid is a web-based platform that provides the hosting, deployment, and management of CCS applications [51]. Scientists or CCS project managers can deploy server images for their applications onto different cloud platforms or onto local server resources and also register their application client images with CitizenGrid so that they are available to end-users (volunteers) who want to participate in CCS projects.

For volunteers, CitizenGrid provides a CCS application portal, where they can discover and participate in CCS applications by launching application clients on their local machine, or where applicable, on a remote cloud platform, for example, OpenStack [24] or the Amazon EC2 cloud infrastructure [25].

While CitizenGrid is intended to be a generic platform, complex citizen science applications such as VAS can be better supported and can provide an enhanced user experience by adding custom application-specific extensions to CitizenGrid. The integration of VAS, LiveQ, and CitizenGrid provides enhanced accessibility and usability of the VAS game, building on the capabilities of LiveQ and CitizenGrid to support deployment, management, and the integration of VT and computing.

In the following sections, we provide an overview of the integration process used to link VAS, LiveQ, and CitizenGrid and present implementation details of the resulting platform, its potential use cases and related work.

IV. Platform Integration Overview

We refer to the integration of the VAS citizen science application [7], the LiveQ job distribution framework [7], [8], and the CitizenGrid [6], [51] platform as VAS-CitizenGrid. In this section, we provide an overview of the structure of the integrated environment then detail the participation and collaboration that needs to be undertaken by individuals working within the VAS-CitizenGrid environment.

The VAS game provides an interface for individuals to learn about particle physics, but it requires underlying computation to support this process. While volunteers who want to participate only from a VT perspective can visit the game website and begin working through the game and learning about the physics aspects of the "Virtual Collider," their interactions will trigger the running of computations and these need to be carried out somewhere. These computations could be carried out on central resources run by the game's operators, but these computations can be large and this is an unsustainable approach as user traffic grows and more people want to take part. Other options might, for example, include using infrastructure-as-a-service cloud resources to support the scaling of resource capacity. However, using such publicly available platforms requires that computation is paid for on a per-use basis. For an educational tool that is freely accessible, this is also not a practical solution. VC is an ideal solution for such a task.

VAS uses the LiveQ job distribution framework to enable and manage the distribution of jobs to the computing resources of volunteers. LiveQ deals with the process of registering users' resources and keeping track of them so that tasks can be farmed out to these resources when user interactions within the VAS game generate them. This explains the reasoning for the integration of VAS and LiveQ, however, there is still a challenge. Volunteers who want to provide their computing resources need a way for these resources to be enabled to

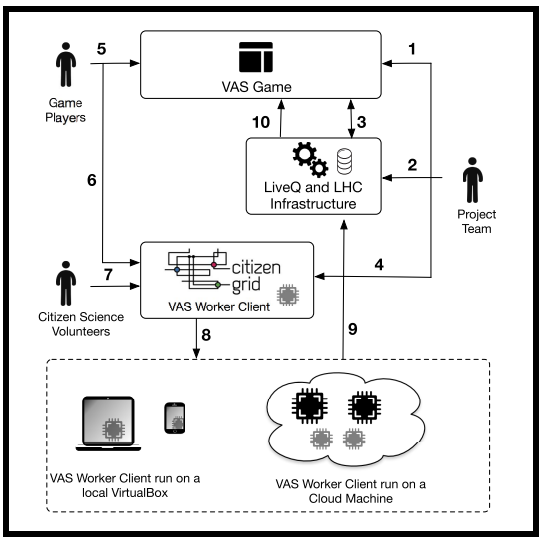

Fig. 6. VAS-CitizenGrid collaborative platform. Numbers represent the participation and interaction steps.

run the VAS simulation software, which can undertake the jobs that are farmed out by LiveQ. In the case of some VC projects, the computation done on volunteers' computers is relatively simple. This does not necessarily mean that the software is not computationally intensive, but it does mean that the code is sufficiently simple that a single small executable, perhaps with a small number of dependencies, can be downloaded to a volunteer's resource and run. This generally makes distributing the code very straightforward. In the case of the VAS simulation software, the code is rather more complex requiring a specific operating system and stack of dependencies and complex configuration. The CernVM virtual machine appliance [26] was developed specifically to address this challenge and is described in more detail in Section V-C. For volunteers to make use of the CernVM virtual machine, it must be deployed and run on their resources and configured to tell the LiveQ infrastructure to which group a user is a member of. This ensures that only computations belonging to that group are distributed to the user's resource(s). This is where CitizenGrid provides an ideal environment to link users with the integrated VAS and LiveQ framework. Users register with CitizenGrid, which provides group management allowing users to form themselves into groups that can then be used within the VAS game. CitizenGrid also provides the ability to host the CernVM machine image and deploy and run this on a user's local machine in a straightforward manner. This can be done using either CitizenGrid's legacy setup for starting VirtualBox virtual machines based on Java Web Start technology or using the CernVM Web API [5] that provides an enhanced method of starting and managing virtual machines running in VirtualBox. CernVM Web API has been integrated within CitizenGrid, which allows the enhanced management of CernVM virtual machines running in VirtualBox from within the CitizenGrid web application.

The linking of these three platforms, therefore, provides a full end-to-end system allowing end users to find the VAS application within the application directory in CitizenGrid (see Fig. 5), join a group or team, and then start one or more virtual machines capable of running the required CERN software stack to enable these machines to undertake simulations of data used in the VAS game. Once a virtual machine is running, it registers with the LiveQ infrastructure which then adds it to the list of machines available to undertake computations for the target group. The volunteer may then play the VAS game themselves or provide their compute resource(s) for others in the same group who are interacting with the game. When computations are generated by game players in this group, they are farmed out to available resources and run, providing feedback to game players within the VAS user interface.

Fig. 6 shows the ordering of the interactions between VAS, LiveQ, and CitizenGrid. We now describe the different participants within such an integrated environment and how they interact with each other. While these descriptions focus on the VAS-CitizenGrid exemplar, they would also apply in the context of other integrated VT and VC applications.

A. Project Participants

A CCS project, such as VAS, that uses integration with CitizenGrid and LiveQ to provide its operational platform to make games available and deploy VC tasks, involves a number of participants. We categorize these participants as project creator teams, game players, and CCS volunteers.

1) *Project Creator Team:*
   a) *Scientists:* Scientists define scientific game requirements, specifications, descriptions, and designs. They provide scientific data, procedures, and the methods that support VC tasks.
   b) *Educators:* Educators develop the educational material based on the scientific processes involved and refine game designs from the learning and creativity perspective.
   c) *Developers:* The developers may include individuals from a range of different development roles—server-side/back-end developers, mini-game developers, and UI developers. These developers implement and test the various software components of the game and the VC tasks that will be deployed to volunteers' computers.
   d) *CitizenGrid Application Provider:* The application provider is the individual in the project creator team who registers the project with the CitizenGrid environment by providing project details and uploading the project's VC client virtual images (worker nodes) that will be downloaded by volunteers' computers.
2) *Thinking Volunteers—Game Players:* These are the individuals who play a scientific game in order to learn about both the scientific processes or domain represented in the game and to support the game operators in undertaking some scientific task.
3) *Computing Volunteers—Citizen Science Volunteers:* Computing volunteers contribute to the CCS project by donating their computing resources. This is achieved by downloading and running VC client images on

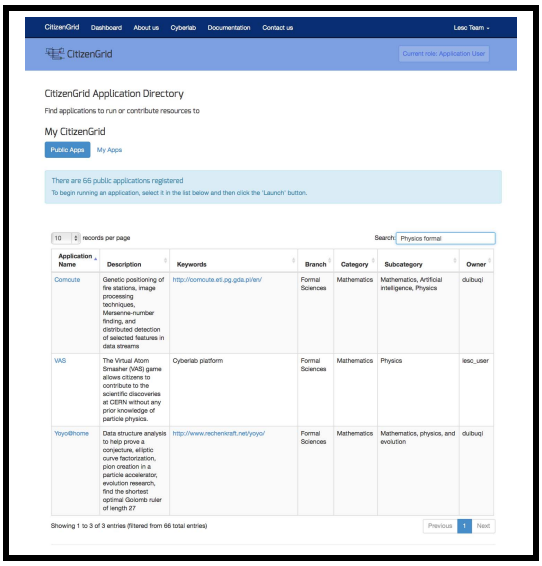

Fig. 7. CitizenGrid application directory showing an example of available CCS projects that users can choose to participate in, including the examples of third-party publicly available CCS projects from other groups.

their local computer. Alternatively, they may choose to support the project by providing their own cloud infrastructure resources or funding remote, public cloud resources, that run the project's VC images (see Fig. 7).

Note that an individual may fall into more than one of the above roles when participating in an integrated VT/VC project.

B. Participation and Interaction Steps in the VAS-CitizenGrid Platform

We now look at the steps shown in Fig. 6 detailing the collaboration between the project creator team, game players, and citizen science volunteers. The steps in the diagram are numbered and the corresponding descriptions are provided as follows.

1) The project team creates and deploys scientific learning games.
2) The project team sets up a VC infrastructure using LiveQ for the real-time distribution and management of VC tasks.
3) The project team configures game and VC components to interact with each other.
4) The project team registers/hosts the VC client images (worker nodes) on the CitizenGrid platform.
5) The game players register their team with the VAS game via the VAS web-based interface and are assigned a team identifier.
6) The game players create a team with their VAS team identifier on CitizenGrid.
7) The volunteers register with the CitizenGrid platform and join a team.
8) The volunteers participate in the VC project by downloading the VC worker node to their local computer and running it via the VirtualBox virtualization software, or by running a worker node on a cloud infrastructure.

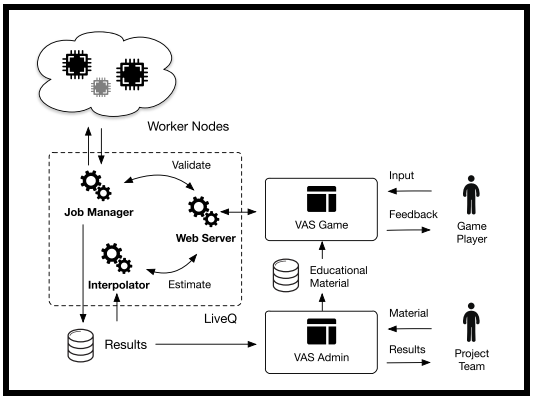

Fig. 8. VAS game and LiveQ interactions.

9) A running worker node receives tasks directly from the VC infrastructure (server).
10) As a game player interacts with the game interface, the VC server updates the interface with new game information and generates new VC tasks to be sent to volunteers' computers for processing.

V. Implementation

The particular scenario of integrating VT and VC is enabled in the system described in this paper by linking the CitizenGrid, LiveQ, and VAS platforms. As in Section IV, we again refer to CitizenGrid-VAS as the collaborative platform that combines the functionalities of CitizenGrid, LiveQ, and VAS. By replacing the VAS game with other CCS games or applications which require real-time interaction between VT and VC tasks, the platform has the ability to offer support for a variety of CCS use cases. In this section, we provide technical details of the integration of the platform components in the CitizenGrid-VAS environment.

A. Integration Between the VAS Game Interface and LiveQ

The integration of the VAS game and the LiveQ framework is shown in Fig. 8. When the game player chooses a parameter and clicks on the "validate the parameter" button, the back-end functionality requests the execution of a Monte Carlo simulation. These simulations will be carried out by one or more distributed computer nodes. This request is routed through the LiveQ framework, which handles the control of the connected VC resources and overseeing the simulation process. The framework is designed in such a way to minimize the overall run time and to keep the user informed throughout the simulation process, providing as much information as possible. This is an important requirement to keep the overall experience interactive since every simulation can take up to half an hour to complete.

The LiveQ framework [8] uses a multivariate interpolation mechanism to estimate the outcome of the simulation and sends a "guess" to the user within a few seconds of their request. In the meantime, it dispatches the job to as many VC worker nodes as possible. The LiveQ framework is fully aware of the entire network of worker nodes, and therefore,

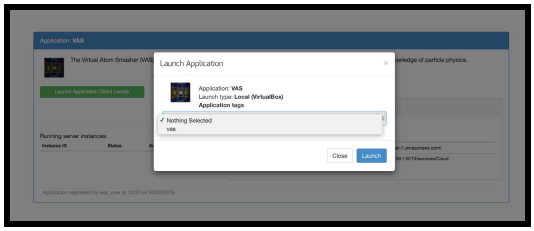

Fig. 9. CitizenGrid interface showing the VAS worker node launch using the team-alias.

it can pick the most appropriate ones or force some of them to discard their current work and start a new task. The job-manager component communicates with the worker network using the Jabber/XMPP protocol that offers flexibility and scalability, even in environments with a slow network or firewall restrictions.

The simulation process used in VAS is highly parallelizable and it is very easy to merge results as they arrive from different worker nodes meaning that the process can easily be handled by multiple independent, distributed computing resources. This is particularly important since the use of real scientific data sets means that significant computation may be required and this can be speed up through the use of larger numbers of compute nodes. Therefore, after a fixed number of events, each worker node sends back its intermediate results, gradually optimizing the results presented in the GUI. When all the workers have completed their tasks, their results are merged into a final results record. The LiveQ framework also takes care of comparing these results against the experimental results and giving a "Goodness of fit" score [27]. The LiveQ framework calculates the X^2 test score between the histograms produced by the simulation and the histograms obtained by the experiments. Therefore, it is possible for the system to automatically understand if the user has succeeded in finding a good solution, and to give the appropriate credit.

B. Integration Between CitizenGrid and VAS

CitizenGrid [6] has been implemented using the Django framework [28]. CitizenGrid is intended to provide a unique one-stop environment for volunteers to search for their favorite VC and gaming projects and to set up and manage game teams through which they would like to contribute their resources. For example, a volunteer signs up to the CitizenGrid platform in order to begin taking part in a CCS project. For team-based projects, the user needs to select their chosen project and become a part of a team. There are two possible approaches in doing this: either they find and select their favorite project and then join a team that is already participating in that project or they first join a team and then select a project to participate in from those that the team is involved with. This creates a resource pool for the team. The advantage of contributing resources through CitizenGrid is that volunteers have an option of contributing their resources to many projects at the same time and can use a common environment to manage their project participation. In addition to their local resources, volunteers can also contribute resources from a public cloud infrastructure.

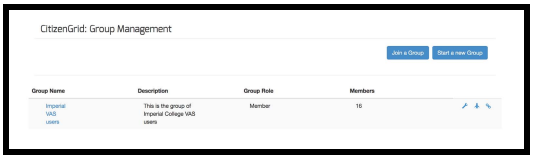

Fig. 10. CitizenGrid "Group management" interface showing the list of existing groups and options of starting a new group and joining an existing group owned by another volunteer.

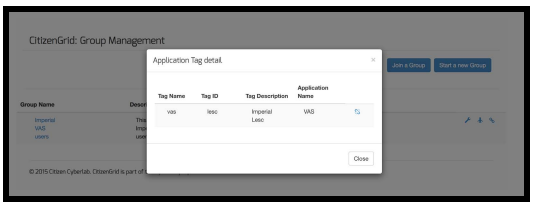

Fig. 11. CitizenGrid "group management" interface showing the VAS application "attachment" to the group using the team-id (labeled as Tag ID).

In the case of the VAS game, each player belongs to a team and the team uses computing resources donated to it for running simulations. The game players use CitizenGrid as a middleware for requesting the computing resources for their team from the volunteers that are registered with CitizenGrid. VAS and CitizenGrid link game players' worker nodes to volunteers' resources using a *team-id*. This is a unique identifier for a team that is shared between game players and volunteers. CitizenGrid provides a feature for game players to create a group and then attach the VAS application to this group using their unique VAS *team-id* and *team-alias* (see Figs. 10 and 11). Once this has been done, other volunteers joining the same group can see the group's details including the attached VAS application. When volunteers launch the VAS application, they see a drop-down menu for selecting a *team-alias* as shown in Fig. 9. Selecting a team alias from the list results in the contextualization of the VAS worker node (see Section V-C), so that it registers itself as belonging to the specified team and is, therefore, made available to undertake computations for that team. Fig. 12 shows the VAS interface displaying details of a team including the list of team members and the credits they have earned.

The integration of VAS and CitizenGrid was enabled through the addition of group creation and management features that are compatible with the approach used by VAS. We consider that these features are a valuable addition to CitizenGrid since they are likely to be applicable to other VC and VT applications that providers may want to register with CitizenGrid.

C. Integration Between CitizenGrid and LiveQ

Because of the complex stack of dependencies that the experimental software run by VAS requires, it is not possible to cross-compile the software for every operating system. Therefore, the VAS project team decided to use virtualization for worker nodes, using the micro-CernVM [26] as the base virtual appliance. This small virtual machine image provides the basis for a tailored OS distribution for VC projects because

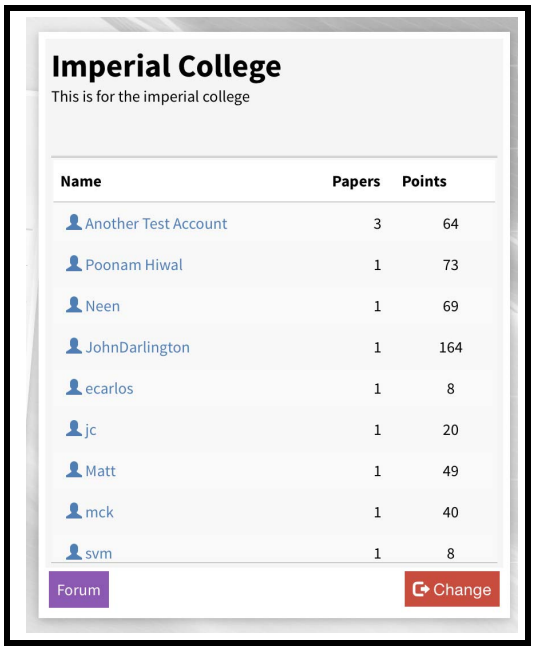

Fig. 12. VAS interface showing game players within a team and the credits earned by the computation carried out by the CitizenGrid team's computing resources.

it minimizes the overall data footprint by transferring the absolute minimum required data over the network. Micro-CernVM is derived from the full CernVM [26] virtual appliance, which is a well-established distribution for experimental software. One of the big advantages for using CernVM is its file system (CVMFS) [29] that offers a high-performance remote file store providing a reliable way of delivering software and data to distributed computer nodes. Nodes mount the CVMFS filesystem and file data are only transferred across the network to the compute nodes when files are requested. This ensures that the absolute minimum required data are transferred across the network to the compute node. Finally, CernVM does not require modifications to the base image to run a given application. Instead, it uses a "contextualization" mechanism to define its boot behavior. This contextualization process determines the functionality that a CernVM instance will provide when it starts up. The required software, already deployed in the CVMFS, can be pulled to the instance at startup time. The worker nodes make use of the LiveQ agent scripts that are already available in CVMFS, and are directed to provide resources for a particular group through the contextualization process. CitizenGrid allows different project teams to advertise, host, and deploy worker nodes for their VC applications. The project team registers the VAS project within CitizenGrid and uploads a VAS contextualized micro-CernVM as the image to be used for the VAS worker node. CitizenGrid has been extended to include the option to personalize worker nodes by automatically embedding the *team-id* into the contextualization process that takes place when starting a VAS compute node.

VI. Usage Scenarios

Through the integration of the CitizenGrid, LiveQ, and VAS platforms, we have demonstrated how VC and VT can be integrated within a single application. This integration benefits both the users of the VAS game and the community of VC providers who can join groups through CitizenGrid and target their computing power to help the corresponding teams within VAS. However, this is just one example of how the integration of these tools can help volunteers contributing both thinking and computational power. In this section, we look at three different usage scenarios facilitated through the integration of the tools described in this paper. In particular, we look at examples of how the integration of CitizenGrid and LiveQ can provide a generic platform for managing applications, groups of volunteers, and the distribution of computational tasks to computing volunteers.

A. Aggregation of Donated Resources

In citizen science, volunteers can choose different ways to contribute their knowledge or computing resources to a wide range of available scientific projects. In the case of VC, this requires that VC projects are set up in such a way that they can take advantage of using a number of distributed computational resources to service their computational requirements, for example, their local machine(s) or remote cloud computing resources. The existing VC projects generally rely on volunteers' personal computers and do not make use of cloud computing infrastructure. However, CitizenGrid offers volunteers the ability to use their local machine(s) or remote cloud computing resources to undertake VC tasks for a project of their choice. CitizenGrid allows the use of any distributed task management middleware such as BOINC or LiveQ for results aggregation at the application's server. Fig. 13 shows the CitizenGrid interface displaying the application details for the VAS game. It shows two registered cloud images targeting either the OpenStack private cloud or Amazon EC2 public cloud platforms. The user can choose to start one or more instances on one of these platforms depending on what platforms they have access to.

Additionally, by integrating CitizenGrid and LiveQ, we can provide a comprehensive framework for enabling users to offer their resources in a targeted manner and for application providers to efficiently aggregate a potentially large pool of geographically distributed resources to service their project's computational requirements. The group capabilities within CitizenGrid that allow "team-based" participation in projects offer further capabilities for the CCS projects to use team-based approaches and team-focused incentives within their applications. Volunteers may join different teams and contribute their resources to different projects collaboratively or individually giving them flexibility in how their resource capacity is used, via a single interface.

B. Meeting the Demand for Real-Time Computation

In games such as VAS, real-time computation is important to ensure that players receive feedback in a reasonable time when the underlying system is undertaking tasks that are computationally intensive. The VAS game interface could be replaced by a range of other computationally intensive scientific applications which require real-time computing responses,

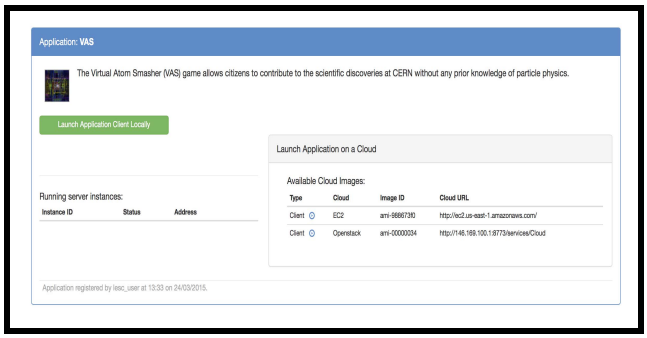

Fig. 13. CitizenGrid: VAS application client details.

e.g., a disaster model mapping application that requires huge computing resources in a short span of time to generate a real-time disaster map based on real-time data to predict and understand the effects of a disaster. Two very popular citizen science human computation projects, Foldit [30], [31] and Eyewire [32], [33], that currently attract up to 4000 volunteers every day, are an ideal example of the challenges of managing the demand for real-time computation. The computational resources required in these games are significant and the projects currently use dedicated servers. In such an environment, handling a sudden increase in demand from users is a major challenge. Game operators face their resources not being able to stand up to the requirements for computation and ultimately crashing under an unmanageable load from the increased user requirements. On the other hand, when usage is low, if a dedicated resource pool is being used to operate the game, resource capacity is likely to be wasted. While cloud computing infrastructure is one possible way of handling changes in demand, this is likely to be costly over a long period of time and impractical for a freely available scientific game. By integrating with a VC environment, game operators can take advantage of contributed resources made available by VC providers. An environment like CitizenGrid offers a way to pool VC providers and build a community of volunteers who can respond when an application has a need for more computing power. This is something that can be challenging to handle in an open environment, where it may be difficult to get access to volunteers' resources at the times when they are needed.

C. Platform for Promoting STEM

The final area where we see the integration of CitizenGrid, LiveQ, and VAS as an ideal model to support other VC and VT applications is in the promotion of science and technology education. The CitizenGrid-VAS platform can be used as a teaching tool in schools and universities for informal learning. In recent years, a number of do-it-yourself open-source game frameworks, such as RedWire [34], have been designed and implemented and these could be used in place of the VAS game to offer alternative game environments to volunteers. Such platforms can be used to design scientific or educational games and can take advantage of the CitizenGrid and LiveQ integrated platform to offer a way to promote their games to potential users and access VC power to handle any computationally intensive aspects of the games. Using such tools, researchers/game developers can also write exciting standalone educational games which can subsequently be integrated with VC projects using the standard interfaces provided by the CitizenGrid and LiveQ platforms.

VII. Related Work

The concept of combining volunteer gaming along with VC, where volunteers donate their computing resources to run simulations or other tasks, has been suggested in the literature [35], [36]. However, these games solve only specific problems and are tightly coupled with volunteers' human computation tasks. In the last decade, the importance of games in human computation (VT) has been confirmed by many studies [37], [38], [52]. There have been a number of human computation games that have been implemented to solve some complex problems [39], for example, the Wildfire Wally game was designed to solve graph search problems [40]. These games are casual games, and there are no scientific and technical learning components associated with them [40]–[43]. However, there is a significant complexity involved in the design of these games to make the most of the potential of human computation [41]. In addition to this, the complexities increase when these games are integrated with VC tasks. In the current literature, only a small number of platforms exist that act as directories to host VC projects [6], [16], [22]. A popular platform with users is the IBM World Community Grid [16], which currently hosts five active projects that use the BOINC [22] middleware for distributed task management. There are a number of client-server-based VC middleware platforms that have been developed in the last two decades, e.g., BOINC [22], Co-pilot [44], Cosm [45], and Bayanihan [46]. A few successful projects, for example, Folding@home [47], [48] and Distributed.net [49], use the Cosm networking libraries for distributed networking. However, the majority of VC projects that have started in the last few years use BOINC [22]. BOINC is a distributed batch processing system. BOINC clients, which run on volunteers' computers, can pull tasks from the BOINC server when they are idle in order to process them locally. The volunteer's computer stores a batch of processed tasks before sending them back to the project server. Co-pilot [44] is another distributed task management system that is designed for CERN distributed computing projects, which acts as a *gateway* between CERN's grid-infrastructure and volunteer resources. Co-pilot allows the distribution of CERN worker nodes onto both cloud infrastructure and volunteers' computers. However, none of the above-mentioned distributed task management systems is designed for real-time interaction, making them unsuitable for an interactive game environment where a player's next-move depends on the outcome of their previous move. To the best of our knowledge, interactive games where real-time VC task distribution and management are combined are not described in the CCS literature. The model of integrating CitizenGrid with the LiveQ framework for handling distribution of VC tasks to volunteers' machines provides the flexibility of integrating a wide variety of scientific games to a distributed

computing infrastructure which requires real-time computation to be carried out by volunteers' resources.

VIII. Conclusion

In this paper, we have presented a CCS deployment scenario which combines VC and thinking tasks within a single project. We have also shown how the three existing platforms CitizenGrid, LiveQ, and VAS can be linked to form an integrated CCS environment. CitizenGrid is a middleware platform for deploying VC clients either using virtual machine images on volunteers' computers or on cloud infrastructures such as Amazon EC2. The VAS is an online particle physics game and LiveQ is a real-time task distribution and management system.

CitizenGrid-VAS, the platform resulting from the integration of these three tools, demonstrates how a game-based collaborative VC platform can be integrated with a citizen science game. The platform allows the easy deployment of a scientific and educational game that requires the use of volunteers' donated computing resources (free CPU cycles) in real time. The CitizenGrid and LiveQ aspects of the integrated platform are flexible and could be used in a variety of other CCS project scenarios. As part of our future work, we are planning to extend this collaborative platform to include the following new functionality: 1) a resource request management interface on the CitizenGrid platform, which allows game players to initiate a request for volunteer resources. Volunteers then decide to choose to provide their computing resources to a particular game player and 2) decoupling of LiveQ, CERN VM, and the VAS game interface. This will provide more flexibility to a game designer to use LiveQ and VAS with other platforms.

Acknowledgment

The authors would like to thank the members of the VAS, LiveQ, and CitizenGrid teams for their support of this paper.

References

[1] M. Haklay, *Crowdsourcing Geographic Knowledge, Volunteered Geographic Information (VGI) in Theory and Practice*, D. Sui, S. Elwood, and M. Goodchild, Eds., 1st ed. The Netherlands: Springer, 2013, doi: 10.1007/978-94-007-4587-2.

[2] P. Yadav and J. Darlington, "Computation platform deployment scenarios," Imperial College London, London, U.K., Citizen Cyberlab (FP7-317705) Project Rep. D3.3, 2014, accessed: Apr. 12, 2015. [Online]. Available: http://citizencyberlab.eu/research/publications-and-reports/

[3] P. Yadav and J. Darlington, "Conceptual frameworks for building online citizen science projects," *J. Human Comput.*, vol. 3, no. 12, pp. 213–223, 2016.

[4] P. Yadav and J. Darlington, "Design guidelines for the user-centred collaborative citizen science platforms," *J. Human Comput.*, vol. 3, no. 11, pp. 205–211, 2016.

[5] I. Charalampidis *et al.*, "CernVM WebAPI—Controlling virtual machines from the Web," in *Proc. 21st Int. Conf. Comput. High Energy Nucl. Phys.*, Okinawa, Japan, Apr. 2015, p. 8.

[6] P. Yadav, J. Cohen, and J. Darlington. "CitizenGrid: An online middleware for crowdsourcing scientific research." Accessed: Aug. 12, 2017. [Online]. Available: https://arxiv.org/abs/1707.09489

[7] I. Charalampidis and P. Skands, "Initial prototype of a virtual atom smasher game," CERN, Eur. Org. Nucl. Res., Citizen Cyberlab (FP7-317705) Project Rep., 2014, accessed: Apr. 12, 2015. [Online]. Available: http://citizencyberlab.eu/research/

[8] I. Charalampidis. *LiveQ: An Interactive Volunteering Computing Batch System Source-Code (GitHub)*. Accessed: May 26, 2015. [Online]. Available: https://github.com/wavesoft/LiveQ

[9] I. Charalampidis. *Virtual Atom Smasher Source-Code (GitHub)*. Accessed: May 26, 2015. [Online]. Available: https://github.com/wavesoft/virtual-atom-smasher

[10] A. Bowser *et al.*, "Using gamification to inspire new citizen science volunteers," in *Proc. 1st Int. Conf. Gameful Design, Res., Appl. (Gamification)*, 2013, pp. 18–25. [Online]. Available: http://doi.acm.org/10.1145/2583008.2583011

[11] C. Jennett and A. L. Cox, "Eight guidelines for designing virtual citizen science projects," in *Proc. Citizen + X, Volunteer-Based Crowdsourcing Sci., Pub. Health, Government, Papers HCOMP Workshop*, 2014, pp. 16–17.

[12] I. Iacovides, C. Jennett, C. Cornish-Trestrail, and A. L. Cox, "Do games attract or sustain engagement in citizen science?: A study of volunteer motivations," in *Proc. CHI Extended Abstracts Hum. Factors Comput. Syst. (CHI EA)*, 2013, pp. 1101–1106. [Online]. Available: http://doi.acm.org/10.1145/2468356.2468553

[13] P. Darch and A. Carusi, "Retaining volunteers in volunteer computing projects," *Philos. Trans. Roy. Soc. London A, Math. Phys. Sci.*, vol. 368, no. 1926, pp. 4177–4192, 2010.

[14] A. Shahri, M. Hosseini, R. Ali, and F. Dalpiaz, "Gamification for volunteer cloud computing," in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2014, pp. 616–617.

[15] Y. Zhao and Q. Zhu, "Evaluation on crowdsourcing research: Current status and future direction," *Inf. Syst. Frontiers*, vol. 16, no. 3, pp. 417–434, Jul. 2014. [Online]. Available: http://dx.doi.org/10.1007/s10796-012-9350-4

[16] *World Community Grid*. Accessed: Apr. 10, 2015. [Online]. Available: http://www.worldcommunitygrid.org/

[17] F. Raoking, J. M. Cohoon, K. Cooke, M. Taufer, and T. Estrada, "Gender and volunteer computing: A survey study," in *Proc. IEEE Frontiers Edu. Conf. (FIE)*, Oct. 2014, pp. 1–5.

[18] T. Estrada, K. L. Pusecker, M. R. Torres, J. Cohoon, and M. Taufer, "Benchmarking gender differences in volunteer computing projects," in *Proc. IEEE 9th Int. Conf. eSci.*, Oct. 2013, pp. 342–349.

[19] *Virtual Atom Smasher*. Accessed: Jun. 20, 2015. [Online]. Available: http://test4theory.cern.ch/vas/

[20] *Professor: Tuning Tool for Monte Carlo Generator*. Accessed: Apr. 10, 2015. [Online]. Available: https://professor.hepforge.org

[21] *Seti@home: An Astronomy Citizen Science Project*. Accessed: Apr. 23, 2014. [Online]. Available: http://seti.berkeley.edu/setiathome

[22] *Boinc Middleware Supported Projects List*. Accessed: Apr. 23, 2014. [Online]. Available: http://boinc.berkeley.edu/projects.php

[23] S. Yi, E. Jeannot, D. Kondo, and D. P. Anderson, "Towards real-time, volunteer distributed computing," in *Proc. 11th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2011, pp. 154–163.

[24] OpenStack. *Open Source Cloud Computing Software*. Accessed: Apr. 23, 2014. [Online]. Available: http://www.openstack.org

[25] *Amazon Web Services (AWS) Elastic Compute Cloud (EC2)*. Accessed: Apr. 25, 2015. [Online]. Available: http://aws.amazon.com/ec2/

[26] *μCernVM: CERN Micro Virtual Machine Image*. Accessed: Apr. 23, 2014. [Online]. Available: http://cernvm.cern.ch/portal/ucernvm

[27] G. K. Smyth, "Pearson's goodness of fit statistic as a score test statistic," in *Statistics and Science: A Festschrift for Terry Speed* (Lecture Notes-Monograph Series), vol. 40. Beachwood, OH, USA: Institute Mathematical Statistics, 2003, pp. 115–126. [Online]. Available: http://projecteuclid.org/euclid.lnms/1215091138

[28] Django. (2014). *Django Software Foundation (2013) the Web Framework for Perfectionists With Deadlines*. Accessed: Apr. 25, 2014. [Online]. Available: https://www.djangoproject.com

[29] J. Blomer, P. Buncic, D. Dykstra, and R. Meusel, "The CernVM file system," CERN, Geneva, Switzerland, Tech. Rep. 2.1-6, CernVM-FS 2.1.20, Mar. 2015. [Online]. Available: https://ecsft.cern.ch/dist/cvmfs/cvmfstech-2.1-6.pdf

[30] F. Khatib *et al.*, "Algorithm discovery by protein folding game players," *Proc. Nat. Acad. Sci. USA.*, vol. 108, no. 47, pp. 18949–18953, 2012.

[31] *Foldit: Online Protein Folding Game*. Accessed: Jul. 10, 2015. [Online]. Available: https://fold.it/

[32] J. S. Kim *et al.*, "Space–time wiring specificity supports direction selectivity in the retina," *Nature*, vol. 509, pp. 331–336, May 2014.

[33] M. SengLab. *EyeWire: The Citizen Science Project*. Accessed: Apr. 23, 2014. [Online]. Available: http://eyewire.org

[34] *Redwire: A Mixing Game Platform*. Accessed: Apr. 19, 2015. [Online]. Available: http://redwire.io/game/list

[35] C. Cusack, C. Martens, and P. Mutreja, "Volunteer computing using casual games," in *Proc. Future Play Int. Conf. Future Game Design Technol.*, 2006, pp. 1–8.

[36] C. A. Cusack, E. Peck, and M. Riolo, "Volunteer computing games: Merging online casual gaming with volunteer computing," in *Proc. Acad. Conf. Meaningful Play*, 2008, pp. 1–34.
[37] M.-C. Yuen, L.-J. Chen, and I. King, "A survey of human computation systems," in *Proc. Int. Conf. Comput. Sci. Eng. (CSE)*, Aug. 2009, pp. 723–728.
[38] A. Bowser, D. Hansen, and J. Preece, "Gamifying citizen science: Lessons and future directions," in *Proc. Gamification Res. Netw. CHI*, 2013, pp. 1–4.
[39] V. Curtis, "Online citizen science games: Opportunities for the biological sciences," *Appl. Transl. Genomics*, vol. 3, no. 4, pp. 90–94, 2014.
[40] E. Peck, M. Riolo, and C. Cusack, "Wildfire wally: A volunteer computing game," in *Proc. Conf. Future Play*, 2007, pp. 241–242.
[41] S. Cooper *et al.*, "The challenge of designing scientific discovery games," in *Proc. 5th Int. Conf. Found. Digit. Games (FDG)*, 2010, pp. 40–47. [Online]. Available: http://doi.acm.org/10.1145/1822348.1822354
[42] D. Toth, "Volunteer computing with video game consoles," in *Proc. 6th WSEAS Int. Conf. Softw. Eng., Parallel Distrib. Syst. (SEPADS)*, 2007, pp. 102–106. [Online]. Available: http://dl.acm.org/citation.cfm?id=1353801.1353820
[43] J. Šimko and M. Bielikov, "Games with a purpose: User generated valid metadata for personal archives," in *Proc. 6th Int. Workshop Semantic Media Adaptation Pers. (SMAP)*, Dec. 2011, pp. 45–50.
[44] P. Buncic and A. Harutyunyan, "Co-pilot: The distributed job execution framework," Portable Anal. Environ. using Virtualization Technol. (WP9), CERN, Geneva, Switzerland, Tech. Rep., Mar. 2011.
[45] *Cosm Software Stack*. Accessed: Jul. 27, 2015. [Online]. Available: http://www.mithral.com/cosm/
[46] L. F. G. Sarmenta and S. Hirano, "Bayanihan: Building and studying Web-based volunteer computing systems using java," *Future Generat. Comput. Syst.*, vol. 15, nos. 5–6, pp. 675–686, 1999.
[47] Stanford University. *Folding Home*. Accessed: Jul. 27, 2015. [Online]. Available: https://folding.stanford.edu/home/the-software/
[48] A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande, "Folding home: Lessons from eight years of volunteer distributed computing," in *Proc. Int. Workshop High Perform. Comput. Biol. (IPDPS)*, May 2009, pp. 1–8.
[49] *Volunteer Distributed Computing Platform*. Accessed: Jul. 27, 2015. [Online]. Available: http://www.distributed.net/
[50] VirtualBox. Accessed: Jul. 27, 2015. [Online]. Available: https://www.virtualbox.org/
[51] *CitizenGrid—Open Source Github Repository*. Accessed: Jan. 5, 2016. [Online]. Available: https://github.com/ImperialCollegeLondon/citizengrid
[52] N. Oliveira, E. Jun, and K. Reinecke, "Citizen science opportunities in volunteer-based online experiments," in *Proc. Conf. Hum. Factors Comput. Syst. (CHI)*, May 2017, pp. 6800–6812.

Poonam Yadav (GS'09–M'11) received the M.Tech. degree from the Indian Institute of Information Technology, Allahabad, India, in 2007, and the Ph.D. degree in computing from Imperial College London, London, U.K., in 2011.

She is currently a Research and Teaching Associate with the Computer Laboratory, University of Cambridge, Cambridge, U.K. She has authored over 30 papers in distributed systems, social computing, sensor systems, and Internet of Things.

Dr. Yadav was a recipient of the U.K.–India Education and Research Initiative Ph.D. Award. She was involved in various NERC, TSB, EU, EPSRC, IBM, and Microsoft funded research projects. She is currently the Chair of ACM-W U.K. professional chapter and a member of ACM and BCS.

Ioannis Charalampidis was an OpenLab Fellow with CERN, Geneva, Switzerland, where he was an Architect and Software Engineer with the PH/TH Group. He is the Lead Developer of virtual atom smasher game prototype. He has been a Frontend Engineer with Mesosphere, Inc., San Francisco, CA, USA since 2016. He has authored a number of research papers on CernVM, cloud computing, and virtualization.

Jeremy Cohen received the Ph.D. degree in computer science from Imperial College London, London, U.K.

He is currently a Research Fellow with the Department of Computing, Imperial College London. His current research interests include e-Science and the development of tools, services, and applications to simplify access for domain scientists to a range of different computational infrastructure.

John Darlington is currently a Professor and the Head of the Social Computing Group with the Department of Computing, Imperial College London, London, U.K. He has a long and distinguished track record both in the development of novel software technologies and in the creation of facilities to improve the accessibility and ease of use of computational resources. He was involved in pioneering developments in functional programming languages, program transformation, functional skeletons, co-ordination forms (later adopted by Google as map/reduce), and component-based application development frameworks. His current research interests include the use of cloud computing to support a variety of public and civic Internet services and applications and the economic and social issues surrounding their successful use.

Francois Grey has been an Invited Professor with the University of Geneva, Geneva, Switzerland, since 2014. He is a Co-ordinator of the Citizen Cyberlab, Geneva, a partnership between CERN, the United Nations Institute for Training and Research and the University of Geneva. He is currently a Physicist with the University of Geneva with a passion for citizen science. He has co-authored over 100 scientific publications in peer-reviewed international journals and holds seven patent applications (five granted).