Coordinating Volunteer Computing

David P. Anderson

Space Sciences Laboratory
University of California, Berkeley
Berkeley California USA
davea@berkeley.edu

ABSTRACT

Volunteer computing lets consumers donate the unused processing and storage capacity of their computing devices (desktop, laptop, mobile) to science research projects. It can provide Exa-scale high-throughput computing, and it offers a scalable and sustainable alternative to data-center computing. Since its inception in 2004, BOINC-based volunteer computing has used a "free-market" model in which scientists create and promote projects, and volunteers choose from among these projects. Problems inherent in this model – notably the risk in creating a project – have limited the adoption of volunteer computing. To move beyond these limits, we developed a new model in which volunteers register for science areas rather than for projects, and a central "coordinator" allocates computing resources to projects.

CCS CONCEPTS

10010147.10010919: Computing methodologies, distributed computing methodologies.

KEYWORDS

Volunteer computing, high-throughput computing, scientific computing.

1 Introduction

Volunteer computing (VC) is the use of consumer digital devices, such as desktop and laptop computers, tablets, and smartphones, for high-throughput scientific computing. People participate in VC by installing a program that downloads and executes jobs from servers operated by science projects. About 700,000 devices are currently participating. These devices have about 4 million CPU cores and 560,000 GPUs, and collectively provide an average throughput of 93 PetaFLOPS.

There are currently about 30 VC projects in many scientific areas and at many institutions. The research enabled by VC has resulted in numerous papers in Nature, Science, PNAS, Physical Review, Proteins, PloS Biology, Bioinformatics, J. of Mol. Biol., J. Chem. Phys, and other top journals [1].

Most VC projects use BOINC, an open-source middleware system [2]. BOINC is distributed under the LGPL v3 license and is available on Github. This paper is concerned with the large-scale structure of BOINC-based VC: how scientists and volunteers participate, and how computing power is divided among scientists.

BOINC originally used a "free market" model in which scientists compete for volunteers and computing power. This model has turned out to have shortcomings which stunted the growth of VC. More recently we have designed and implemented a new "coordinated" model in which – although volunteers still have a significant voice – the allocation of computing power is done centrally.

We have implemented the coordinated model in a system called Science United (https://scienceunited.org). This paper presents the motivations for the coordinated model and describes the structure and implementation of Science United.

2 Volunteer computing Models

2.1 The free-market model

In BOINC's original model, scientists create and operate BOINC "projects" consisting of a web site and a job dispatcher. They recruit volunteers by publicizing their project and creating web pages describing their research.

Volunteers discover VC via the publicity of a project P, which takes them to P's web site. This directs them to download the BOINC client software. When the BOINC client starts, the volunteer is shown a list of projects, from which they select P, thus "attaching" the device to P. The volunteer, perhaps at a later time, can

survey and evaluate other available projects. The BOINC client lets volunteers attach devices to multiple projects and control the division of resources among the projects.

The intention of this model was to create a "market" in which scientists compete for computing power by promoting themselves and their research, and in which volunteers periodically evaluate the set of projects and decide, based on their personal values and interests, how to allocate their computing resources.

More generally, our goal was that VC would divide computing power among scientists based on the aggregated knowledge and values of the public (rather than administrative policies). This was inspired by the Iowa Political Stock Market, which used an analogous approach to predicting election results, with the viewpoint that "Markets allocate scarce resources to their most valued use" [3].

However, the free-market model did not achieve these goals. In spite of the prospect of cheap computing power, relatively few scientists created BOINC projects. Some of the reasons for this are inherent in the model. In the free-market model, creating a project is risky: there's a substantial investment [7], with no guarantee of any return, since no one may volunteer. Adding a VC component to a grant proposal adds uncertainty that may weaken the proposal. Secondly, the model requires that projects publicize themselves. This requires resources and skills (media relations, web design, outreach) that are not readily available to most scientists. Finally, retaining volunteers requires having a steady supply of jobs, and the computing needs of many research groups are sporadic.

The free-market model has also not led to the desired volunteer behavior: most volunteers "lock in" to a few projects and don't actively seek out new ones [4]. Furthermore, the volunteer population is declining. Attracting volunteers is a marketing problem. It's difficult to do effective marketing when there are dozens of competing brands (i.e. project names such as SETI@home and IBM World Community Grid).

2.2 Account Managers

One problem with the free-market model is that it's inconvenient for volunteers to browse lots of project web sites. Instead, we wanted to let volunteers browse and select projects on a single "project chooser" web site.

Instead of directly creating such a site, we enabled other people to create them. To this end we added a mechanism called the "account manager architecture". An account manager (AM) is both a web site and an RPC server. Instead of attaching devices directly to projects, volunteers can attach them to an AM. The BOINC client periodically (typically once per day) issues an "account manager RPC" to the AM. The RPC reply contains a list of projects to which the client should attach. The client then issues "scheduler RPCs" to those projects to get and report jobs; see Figure 1.

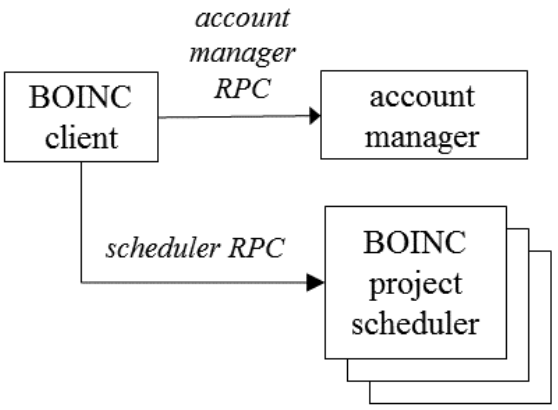

Figure 1: BOINC's account manager architecture.

The AM architecture was used by third-party developers to create three project-chooser sites: Gridrepublic, BAM!, and GRCPool [5, 6, 8]. These AMs did not significantly change volunteer behavior; however, the architecture proved to be useful for other purposes, as we will show.

2.3 The coordinated model

We developed a new "coordinated model" to address the problems of the free-market model: in particular, to eliminate scientists' financial risk in creating BOINC projects, and to eliminates the need for volunteers to evaluate projects. The model involves a central "coordinator", implemented as an account manager. Volunteers interact with the coordinator through its web site. They attach their BOINC clients to the coordinator, which in turn dynamically attaches the clients to projects. This assignment can change over time; a volunteer may compute for a project that didn't exist when they first registered.

The coordinator allocates computing power among a set of "vetted" projects, and may divide power non-uniformly among these projects. Scientists can apply to the coordinator to have prospective BOINC projects pre-vetted. At that point they can be offered a certain amount of computing throughput; this depends on their

science area, their location, and what types of computing devices their applications can use. They can then proceed to create a project, with minimal risk. Projects need not have continuous computing needs.

The coordinated model differs from the free-market model in several important ways. Volunteers no longer directly control the allocation of computing power to projects; they need not be aware of the existence of projects. Therefore, projects no longer need to publicize themselves, or to operate a web site. Their names are no longer "brands". The coordinator can act as a unified brand for VC. Publicity campaigns (mass media, social media, co-promotions, etc.) can refer to this brand, rather than the brands of individual projects. This allows more effective promotion.

The coordinated model doesn't replace the free-market model; the two co-exist. BOINC projects can operate without being vetted by a coordinator, and volunteers can attach to such projects even if they're also attached to a coordinator.

We have implemented a science-oriented coordinator called "Science United" (SU), located at https://scienceunited.org. As part of the SU registration process, volunteers indicate their "science preferences" – which areas of science they do or do not want to support – and their preferences for the location of the research. This aligns with the motivations of most volunteers; support of science goals have been shown to be the major reason for participation in VC [4]. However, other types of coordinators are possible: for example, coordinators that include commercial as well as scientific projects, or that reward volunteers in virtual currency or in-game credit.

3 Volunteer preferences in Science United

3.1 Keywords

As a basis for SU volunteer preferences, we have defined a system of "keywords" for describing jobs. The system has the following structure:

- There are two keyword categories: "science area" and "location" (the geographical or institutional location of the job submitter).
- Keywords form a hierarchy: each level N+1 keyword is a child of a single level N keyword.
- Each keyword has a permanent integer ID, and short and long textual descriptions.

The hierarchy and the descriptions can change over time.

3.2 Keyword preferences

When a volunteer registers with SU, they specify preferences for science areas and locations. A set of preferences maps keywords to {yes, no, maybe}. "No" means don't run jobs with that keyword. "Yes" means preferentially run jobs with that keyword.

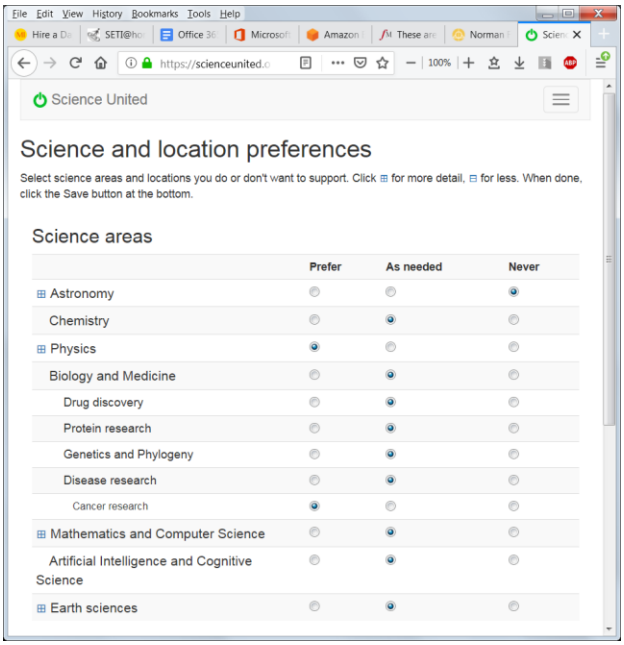

Figure 2: The Science United interface for specifying preferences.

When a new keyword is added, the default setting is "maybe" for all volunteers. Volunteers are notified of the new keyword so that they can change this if they want.

Active SU volunteers average 4.8 "yes" keywords and 0.83 "no" keywords. 87% of the keyword preferences are for science areas; the remainder are for location.

3.3 Project and job attributes

Each project has an associated set of keywords describing its science areas and location. Some projects have applications in multiple science areas or run jobs on behalf of a multiple client institutions; we call these "diverse" projects. For diverse projects, the estimated fraction of jobs having a keyword is associated with the keyword. The set of project keywords can change over time, reflecting changes in the project's workload.

Volunteer preferences may be enforced at the project level. If a project has a keyword with job fraction 1,

and a volunteer has specified "no" for that keyword, the volunteer's devices may not be attached to that project.

For diverse projects, preferences must be enforced at the job level; a volunteer may be willing to run some jobs but not others. For such projects, jobs have an associated set of keywords, specified in the job submission process. For example, if a job is submitted by a cancer researcher at UC Berkeley, the attributes would include "cancer research" and "UC Berkeley".

In this case, volunteer preferences are enforced by the project's BOINC job dispatcher. In deciding which jobs to send to a device, the dispatcher computes a "score" for each job that includes a number of different factors; it then sends the highest-scoring jobs. We extended this to include keywords. For each of the job's keywords, if the volunteer has "yes" the score is incremented, and if "no" the job is not sent.

4 Dividing computing power

The central function of SU is to divide computing power among projects. It does this by assigning projects to volunteer devices. These assignments can change each time the device issues an AM RPC (typically once per day).

The assignment policy has several goals:

- To honor volunteer keyword preferences by preferentially assigning projects with the volunteer's "yes" keywords.
- To allow projects to be allocated different shares of the resource pool (see below).
- To maximize total throughput. For example, if a host has a GPU, it should be assigned at least one project that can use the GPU.

These goals are possibly conflicting; for example, a project with a large share may have keywords with few "yes" preferences. The policy should balance these conflicting goals.

This section describes the factors in more detail, and concludes by describing the assignment policy.

4.1 Platforms and processing resources

A project may not be able to use all volunteer devices. Each device supports one or more "platforms" (Windows/x64, Mac/x64, Linux/ARM, etc.) and has a set of "processing resources", including a CPU and possibly one or more GPUs of various vendors (NVIDIA, AMD, Intel). In addition, a device may have virtualization software (VirtualBox) installed.

Each BOINC project has a set of "app versions", each of which runs on a particular platform, uses a specific set of processing resources, and may require VirtualBox. Depending on its app versions, a project may not be able to use a device at all, or it may be able to use only a subset of the device's processing resources.

When an account manager such as SU instructs a BOINC client to attach to a project, it can specify a set of processing resources for which the client should request work. Thus, for example, SU can tell the client to get CPU jobs but not GPU jobs from the project.

4.2 Project shares

SU allows some projects to be given more computing resources than others.

Let M(P) denote the maximum possible rate of computing for a project P, given SU's current resource pool. M(P) is determined by P's keywords and applications. P can use a device D only if P's keywords are compatible with the preferences of D's owner, and it can use D's processing resources (CPU and GPUs) only if it has appropriate applications. Thus M(P) can vary widely between projects.

In SU, each project P has a "share" S(P). Shares are assigned administratively (see Section VI), and may change over time. Roughly speaking, S(P) determines how much computing is available to P compared to other projects with similar M(P), over a time scale on the order of 1 week.

4.3 Resource usage accounting

SU does accounting of processing resource usage. This serves several purposes: it provides a basis for enforcing project shares, it gives an estimate of the system-wide throughput, and it provides basis for volunteer incentive such as graphs of work done recently, work "milestones", and so on.

BOINC has a sophisticated credit system for estimating the FLOPs performed by completed jobs. It is fairly "cheat-proof": it is difficult to get credit for computation not actually performed. However, the system is based in part on job replication, so credit for a job may not be granted until the companion job is completed, which could take weeks. This makes it unsuitable for SU's purposes.

Instead, SU uses a quantity called "estimated credit" (EC), which is maintained by the BOINC client on a per-job and per-project basis, based on the runtime of jobs and the peak FLOPS of the processors they use.

EC is a cruder estimate than credit, and it is not cheat-proof. But it accumulates continuously, with no need to wait for job completion or validation.

4.4 Share-based prioritization

SU enforces project shares by prioritizing projects that haven't used their share of resources recently. It maintains, for each project P, its average rate of computing over the last week, A(P). We then let

$$A_{frac}(P) = A(P) \Big/ \sum_{projects\ Q} A(Q)$$

$A_{frac}(P)$ is the fraction of total computing done by P. Similarly, let

$$S_{frac}(P) = S(P) \Big/ \sum_{projects\ Q} S(Q)$$

$S_{frac}(P)$ is P's fraction of the total share. We then let

$$E(P) = A_{frac}(P)/S_{frac}(P)$$

E(P) represents the excess computing that P has received, relative to its share, over the last week. It is used to prioritize projects in the assignment algorithm (see below). At any point, computing resources are preferentially assigned to projects P for which E(P) is least.

This model handles both continuous and sporadic workloads well. For a project P with sporadic workload, E(P) will usually be near zero. When P generates a burst of work, it will have priority over the continuous-workload projects, and the work will get done quickly.

When a computer is assigned to a project, there will be a delay of about a day (the client polling period) until computation is reported to SU by clients. This means that the same project (the one for which E(P) is least) will be assigned to all hosts during that period. This is undesirable. To solve this problem, we dynamically adjust A(P) by an appropriate amount when a computer is attached to or detached from P. At the end of each accounting period, A(P) is reset based on the accounting history.

4.5 Preventing device starvation

When SU assigns a project to a device, it assumes that the project can supply jobs that use the given processing resources. But this may not be the case. The project may be temporarily down, it may not have jobs using the target resources, or its jobs that use the target resources may have keywords disallowed by the user's preferences. This can lead to device starvation (idle device instances).

To address this problem, the BOINC client keeps track of (*project*, *resource*) pairs that are temporarily starved: that is, for which the last scheduler RPC requested work for the given resource, but none was returned. The list of temporarily starved (*project*, *resource*) pairs is included in the AM RPC request, and is used by the assignment algorithm (see below) to prevent device starvation.

4.6 Assignment algorithm

BOINC clients using SU periodically (once per day) issue an AM RPC. The request message includes a list of currently-attached projects and their CPU and GPU EC totals; these are used to update accounting records. The reply message includes a list of projects to attach to. The client detaches from any projects not on this list, after completing pending jobs. For each project, the reply specifies a "resource share": a value of zero means that the client should do work for this project only if none of the other projects have work available.

Being attached to a project has a disk overhead; the client caches applications files for the project, which may include large VM image files. Hence we want to limit the number of projects to which each client is attached. On the other hand, if a project has a large disk footprint on a client, we may want the client to remain attached, with a zero resource share, so that files don't need to be downloaded again the next time the project is attached.

With these factors in mind, here is a sketch of the project assignment algorithm currently used by SU:

First, we discard projects that can't be used, either because of keyword preferences or because the project doesn't have app versions that can use the device. We compute a "score" for each remaining project. This score is the weighted sum of several components:

- A keyword factor. Increment the score if the project has keywords in the volunteer's "yes" list
- Subtract the project's allocation balance E(P).
- Increment the score if the host is already attached to the project.

The weights for each of these terms have been chosen empirically.

Then, for each processing resource R, we find the highest-scoring project that can use R. This is the set

of projects to be attached. If the client is currently attached to a project not in this set but whose disk footprint exceeds a threshold (currently 100 MB) and whose score is nonzero, we tell the client to remain attached with zero resource share.

5 Implementation and status

SU is implemented in PHP. It uses a MySQL database to store volunteer and project information, accounting data, and so on. The SU source code is distributed under the LGPL v3 license and is available on Github.

SU required some modifications to the BOINC client, such as the starvation monitoring described in section IV. These changes are in the current client release (7.14).

Science United was launched in 2018. Currently it has about 1400 volunteers and 1800 computers, of which 1500 have usable GPUs. These computers process about 50,000 jobs per day and have a throughput of roughly 200 TeraFLOPS. Figures 3 and 4 show recent throughput histories for CPU and GPU respectively. Projects such as Rosetta@home appear only in the CPU graph because they have no GPU app versions.

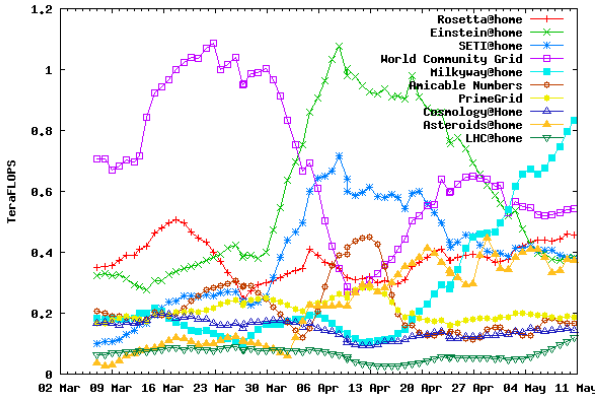

Figure 3: CPU throughput of the top projects over the last two months.

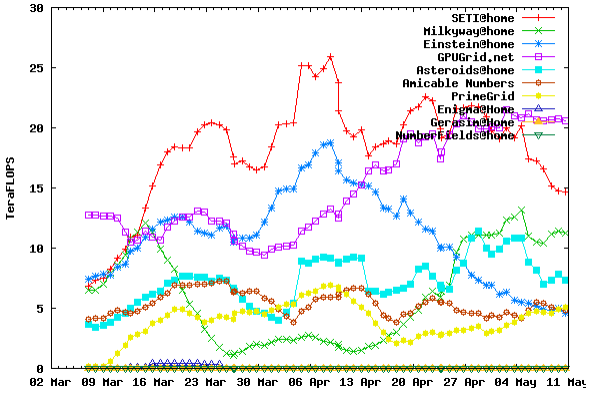

Figure 4: GPU throughput of the top projects over the last two months.

5.1 Administration and policies

We plan to establish a committee to determine coordinator policies, including project vetting and resource allocation. The committee may include representatives of scientific funding agencies, leaders of the coordinator project, and members of the volunteer community. The committee will decide what projects to vet, based on criteria such as:

- The project's computing is directed toward a scientific or technical goal (broadly interpreted to include things like mathematics and cryptography).
- The project is non-commercial.
- The project's leadership has a certain level of qualification (as demonstrated, e.g., by publications).
- The project can prove that it follows various security practices, such as application code-signing on secure offline machines.

The committee will define a process by which potential new projects can apply for vetting. A scientist or organization could apply for vetting, then submit a grant proposal to fund the development of the project. The committee will assign shares to vetted projects, based on need, merit, or other factors.

6 Future work

6.1 Throughput guarantees

In some existing HTC systems, a user can be guaranteed a minimum throughput over a given period of time with high probability. Can we offer analogous guarantees with VC resources?

The performance of a pool of volunteer computers varies over time, in terms of both throughput and job latency. However, with a large pool, these quantities change slowly, and we can establish the statistics of this change. For example, given the total throughput T at a given time, we could find a $T_0 < T$ such that total throughput will remain above T_0 for a week with a given confidence level.

Similarly, given a Science United resource pool and a particular set of projects and shares, the throughput seen by a project should remain fairly constant over time. These throughputs can be manipulated, within limits, by changing shares.

How can we predict, given a particular set of project shares, how much throughput each project will get? This depends on many factors: app versions, keywords, the project assignment algorithm, and so on. It's unlikely that it can be determined analytically. Instead, we plan to implement an emulator that does a trace-based simulation of the entire SU system, using the code of the RPC handler, and predicts the throughput of each project. Using this emulator we will be able to compute a mapping from project shares to project throughput, and to find project shares for which a particular project achieves a given throughput. This will provide a basis for guaranteeing throughput to projects over fixed periods.

Such guarantees would be project-level. Can we provide performance guarantees to a particular job submitter within a project that serves multiple job submitters? This is more complex but it may be possible. The BOINC server software allocates resources among computing job submitters within the project. It's possible that the combination of a project-level allocation and a submitter-level allocation can provide some form of performance guarantee to the submitter.

7 Conclusion

We have explained the problems with BOINC's original "free market" framework for volunteer computing and have described the coordinated model and its implementation in Science United. We hope to increase the resrouce pool to the point where its throughput is comparable to data-center HTC providers – perhaps tens of PetaFLOPS. At that point the key goal of the coordinated model – eliminating risk to prospective new projects – will be realized, hopefully resulting in a broader adoption of volunteer computing.

ACKNOWLEDGEMENTS

The idea of coordinated volunteer computing arose from a conversation with Mislav Malenica.

This work was supported by the National Science Foundation, award #1664190.

REFERENCES

[1] "Publications by BOINC projects", https://boinc.berkeley.edu/wiki/Publications_by_BOINC_projects, 2020.

[2] D. P. Anderson, "BOINC: A Platform for Volunteer Computing", Journal of Grid Computing, Nov. 2019.

[3] Forsythe, Robert; Nelson, Forrest; Neumann, George R.; Wright, Jack. "Anatomy of an Experimental Political Stock Market". The American Economic Review, 1992, pp 1142-1161

[4] O. Nov, O. Arazy and D. Anderson, "Technology-Mediated Citizen Science Participation: A Motivational Model," in Fifth International AAAI Conference on Weblogs and Social Media (ICWSM 2011), Barcelona, 2011.

[5] Gridrepublic: http://gridrepublic.org/

[6] BAM!: http://bam.boincstats.com/

[7] D. Kondo, J. Bahman, P. Malecot, F. Cappello and D. Anderson, "Cost-Benefit Analysis of Cloud Computing versus Desktop Grids," in 18th International Heterogeneity in Computing Workshop, Rome, 2009.

[8] Gridcoin, "The Computation Power of a Blockchain Driving Science and Data Analysis", https://gridcoin.us/assets/img/whitepaper.pdf, 2018