Volunteer computing with BOINC

Architecture, case of study analysis

Ian Ribalta Gené

Distributed computing

30/10/2022

Table of Contents

Index of Figures

Index of Tables

Introduction

In this project we will be discussing Volunteer Computing with a case of study of *Berkeley Open Infrastructure for Network Computing* (BOINC).

BOINC as defined by *Berkeley* is an “open source middleware system for volunteer- and grid computing”, and its intent is “to make it possible for researchers to tap into the enormous processing power of personal computers around the world”.[1]

A brief historical introduction about BOINC

To learn what BOINC is, it is good to know its origins.

BOINC comes from the natural evolution of *SETI@home*. SETI was first conceived in January 1995 by *David Gedye* and launched in 1999. During the same time *Gedye* and *David P. Anderson* discussed forming an organisation for SETI-like projects, which they planned to name it Big Science, but it ended up becoming BOINC.[2]

As time went on *SETI@home* needed a separate software platform, and so BOINC was born in 2002.[2]

The different projects that have worked on BOINC are:

Active	• climateprediction.net • Collatz Conjecture • Cosmology@Home • DENIS@Home • Einstein@Home • Gerasim@home • GPUGRID.net • iThena • LHC@home • MilkyWay@home

	• MindModeling@Home • Moo! Wrapper • NFS@Home • NumberFields@home • ODLK • PrimeGrid • Quake-Catcher Network • Rosetta@home • TN-Grid • Universe@Home • World Community Grid (subprojects Clean Energy Project, Discovering Dengue Drugs – Together, FightAIDS@Home, Fiocruz Genome Comparison Project, Help Conquer Cancer, Help Cure Muscular Dystrophy, Human Proteome Folding Project, Help Fight Childhood Cancer, Smash Childhood Cancer) • WUProp@Home • yoyo@home
Beta	• RNA World (beta) • WEP-M+2 Project
Alpha	• nanoHUB@home • RADIOACTIVE@HOME • YAFU
Technology, tools	• BOINC client–server technology • BOINC Credit System • Gridcoin • Charity Engine • GridRepublic • Science United
Completed	• ABC@Home

	• AQUA@home • Artificial Intelligence System • Asteroids@home • BBC Climate Change Experiment • Big and Ugly Rendering Project • CAS@home • Cell Computing • Citizen Science Grid • Correlizer • DistrRTgen • DNETC@HOME • Docking@Home • EDGeS@Home • Enigma@Home • eOn • Evolution@Home (yoyo@home subproject) • FreeHAL • HashClash • Ibercivis • Kryptos@Home • The Lattice Project • Leiden Classical • uFluids@Home • Malaria Control Project • MLC@Home • OProject@Home • Orbit@home • POEM@Home • Pirates@Home • Predictor@home • proteins@home • RALPH@home • Reversi

	• Riesel Sieve (merged with PrimeGrid) • QMC@Home • SAT@home • Seasonal Attribution Project • SETI@home (subproject Astropulse) • SETI@home beta • SIMAP • SLinCA@Home • Spinhenge@home • SZTAKI Desktop Grid • TANPAKU • theSkyNet • VGTU@Home • XtremLab

Boinc Projects[3]

How does BOINC work?

How does the software work

BOINC software works as if it were a single program, although it is made up of several programs[4]. This gives the impression of a single coherent system.

The computers from the project administrators will act as the servers. These computers will have installed the schedulers and data server programs[4].

The computers from the donors will have installed the core client, the applications that do the scientific computing for each project, the GUI, and screensaver if provided by the project[4].
The client software programs communicate via RPCs over a TCP connection[5].

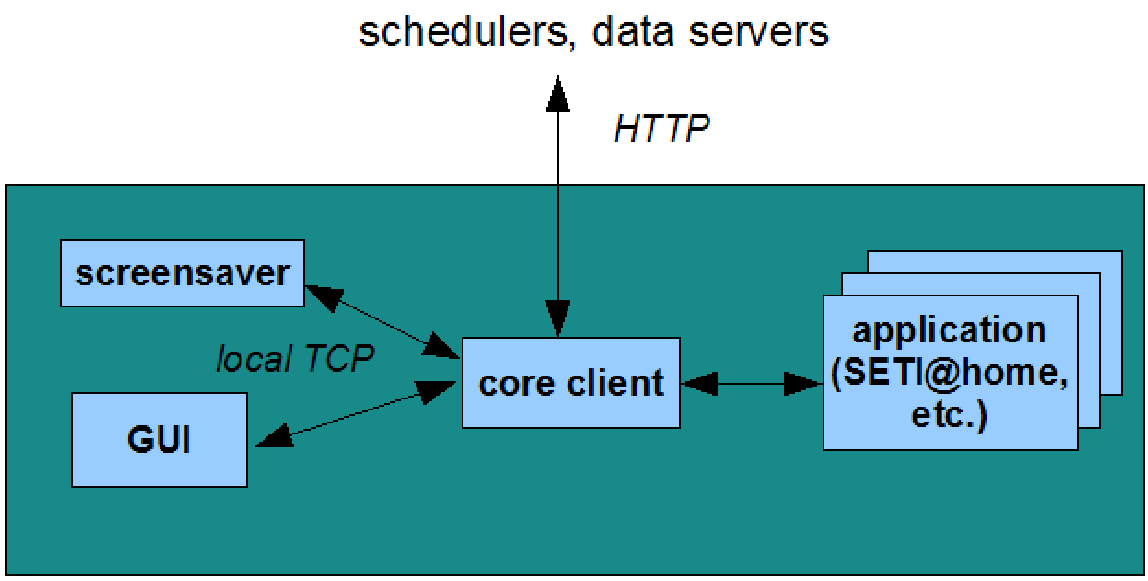

Fig. 1 BOINC software architecture[6]

Credit system, how does it work

Each BOINC project gives credit for the computations. This is a common feature in volunteer computing projects. The credit unit BOINC uses is named Cobblestone, its

value is described by BOINC as “is 1/200 day of CPU time on a reference computer that does 1,000 MFLOPS based on the Whetstone benchmark”[7].
For applying these credit systems there are some basic requirements that not all applications follow. For these type of applications the following formula is used to give the credits $PFC(J) \;=\; \sum r \in R \; runtime\ (J)\ *\ usage(r)\ *\ peak\ flops(r)$[5].

The process of granting credit has the following steps[4] as we can see on Fig. 2:

- A task is sent to two or more computers.
- A computer ends the task and reports a certain amount of credit.
- When two results have returned they are compared. If they are the same the lower of the claimed credits is given.

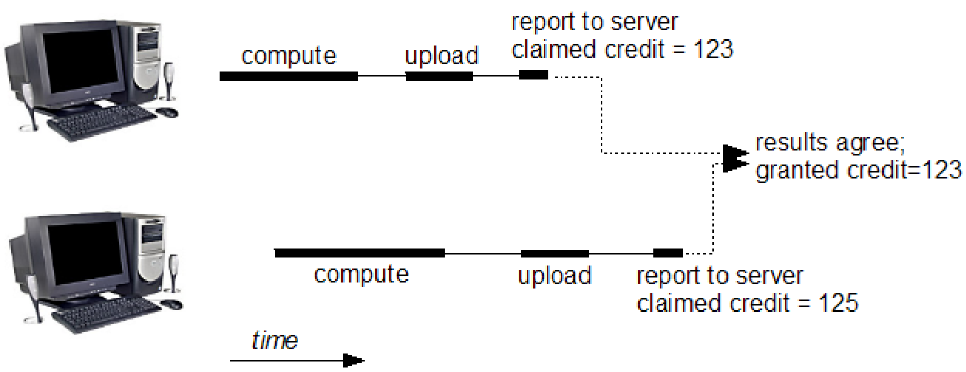

Fig. 2 Credit time diagram[8]

BOINC client-server technology

BOINC has client and server components which make it a centralised architecture. It works as follows: the network communication is initiated by the client that communicates to the project’s task server via HTTP. The request consists of an XML document containing different types of information from the client. The reply consists of another XML document with a list of jobs, a set of data servers from where to retrieve the information, and other files necessary for the computation[9].
On Fig. 3 we can see the architecture of the client-server technology used by BOINC.

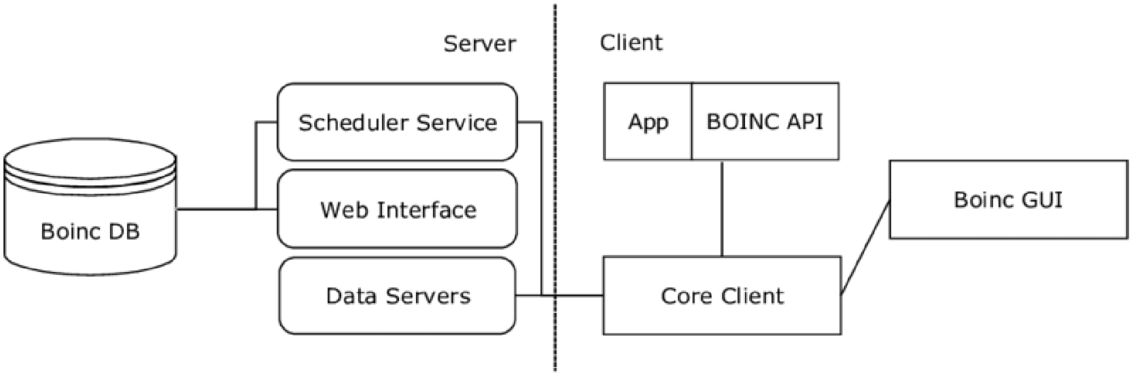

Fig. 3 BOINC Client-Server architecture [10]

Accessibility

When we speak about accessibility, we tend to mean security. In this aspect BOINC has not had any security risk during 16 years [11]. Although this does not mean it is safe, it does show that it is quite secure.

BOINC achieves this using various systems: Code signing the projects, account-based sandboxing, virtual machine-based sandboxing, limited rogue and spoof projects[11].

BOINC is open source, which means you can inspect and modify the code, and build it yourself on your own computer.

Transparency

The only thing you know as a client in BOINC is for which project you are working.

So it has transparency for location, migration and failure.

For the transparency of replication and concurrency you know that another client is working on the same task, for how the credit system works, although the system never tells you if someone else is doing that task or who else is doing it.

The transparency for relocation doesn’t apply in this system as all the tasks will always be done locally.

BOINC does not have access transparency as every time a client wants to work on a new project, a new account needs to be created by the user.

Openness

BOINC works as a middleware, which can work with Windows, Mac and Linux, and with very low hardware requisites[12]. Projects may have higher hardware requisites and not support all operating systems[13].
BOINC supports C++, although if a language uses the C compiler it may work. It can also run Fortran[14].

In conclusion BOINC archives portability, has well-defined interfaces, and the projects may need to be adapted to interoperate with BOINC.

Scalability

As BOINC uses volunteer computing (or Desktop Grid computing), it is not really limited by size as one client can operate from anywhere in the world with access to the internet, without the need for a stable connection.
It isn’t either limited by distance as each node (client) works independently from the others, and the only thing it needs is to be able to download the work unit to do, and once it is done upload it to the project server.
On the administrative side, each project delivers the tasks to the clients with the security features explained on the accessibility section. Each client is able to decide how much computation it gives.

Why use BOINC

Cost comparison

The advantage of volunteer computing is that research projects just need to pay for the server, which costs $100 K per year [5]. Projects of this size average 2 PetaFLOPS throughput per year, by comparison commercial cloud like Amazon EC2 would end up costing $43.53M per year[5]. This huge difference in costs comes from the fact that a research project in volunteer computing does not need to pay for the computation, that means no need to pay for the power consumed by the cooling and the work nodes. It also only needs to store the initial data and the results.

Organisational options

There are various types of organisations that can use BOINC, these are [15]:

- Research group
- Application-centred research community
- Science Gateway
- Institutional umbrella project
- HPC provider

It is also possible to do in-house computing. You just need to create a project, install the server side requisites on a server and the BOINC client on the worker nodes with the created project.

Conclusion

With the research of this topic, we were able to learn how volunteer computing works, and its different applications.

We also got introduced to BOINC, learning that there are more projects in distributed computing apart from *FOLDING@HOME,* that help in a lot of different subjects such as number field practical proof, medical research, and even helping with research from The Large Hadron Collider (LHC) of CERN.

We saw a clever way to deal with one of the problems of centralised architectures, that is unreliable communication. BOINC dealt with it by reducing to the minimum the communications and making sure that the projects got the results, giving the same tasks to various clients and putting a time limit before resending it again to another client in case the results weren’t delivered in time. It also uses TCP to reduce the impact of packet loss.

References

[1] ‘BOINC:About - BOINC’. https://boinc.berkeley.edu/wiki/BOINC:About (accessed Oct. 29, 2022).
[2] J. Koulouris, ‘The Big BOINC ! Projects and Chronology Page’. https://www.angelfire.com/jkoulouris-boinc/ (accessed Oct. 29, 2022).
[3] ‘BOINC client–server technology’, *Wikipedia*. Oct. 16, 2022. Accessed: Oct. 29, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=BOINC_client%E2%80%93server_technology&oldid=1116495697#cite_note-3
[4] ‘How BOINC works - BOINC’. https://boinc.berkeley.edu/w/?title=How_BOINC_works&oldid=5289 (accessed Oct. 29, 2022).
[5] D. P. Anderson, ‘BOINC: A Platform for Volunteer Computing’, *J. Grid Comput.*, vol. 18, no. 1, pp. 99–122, Mar. 2020, doi: 10.1007/s10723-019-09497-9.
[6] ‘Client.png (654×348)’. https://boinc.berkeley.edu/w/images/a/ae/Client.png (accessed Oct. 30, 2022).
[7] ‘Computation credit - BOINC’. https://boinc.berkeley.edu/wiki/Computation_credit (accessed Oct. 29, 2022).
[8] ‘Credit.png (780×278)’. https://boinc.berkeley.edu/w/images/a/af/Credit.png (accessed Oct. 30, 2022).
[9] D. P. Anderson and J. McLeod, ‘Local Scheduling for Volunteer Computing’, in *2007 IEEE International Parallel and Distributed Processing Symposium*, Long Beach, CA, USA, 2007, pp. 1–8. doi: 10.1109/IPDPS.2007.370667.
[10] ‘Fig. 2. BOINC client and server architecture’, *ResearchGate*. https://www.researchgate.net/figure/BOINC-client-and-server-architecture_fig2_228943288 (accessed Oct. 29, 2022).
[11] ‘BOINC Security - BOINC’. https://boinc.berkeley.edu/wiki/BOINC_Security (accessed Oct. 30, 2022).
[12] ‘System requirements - BOINC’. https://boinc.berkeley.edu/wiki/System_requirements (accessed Oct. 30, 2022).
[13] ‘List of projects by system requirements - BOINC’. https://boinc.berkeley.edu/wiki/List_of_projects_by_system_requirements (accessed Oct. 30, 2022).
[14] Nicolás, ‘Answer to “Any experience programming for BOINC?”’, *Stack Overflow*, Dec. 23, 2009. https://stackoverflow.com/a/1950977 (accessed Oct. 30, 2022).
[15] ‘BoincOverview – BOINC’. https://boinc.berkeley.edu/trac/wiki/BoincOverview (accessed Oct. 29, 2022).