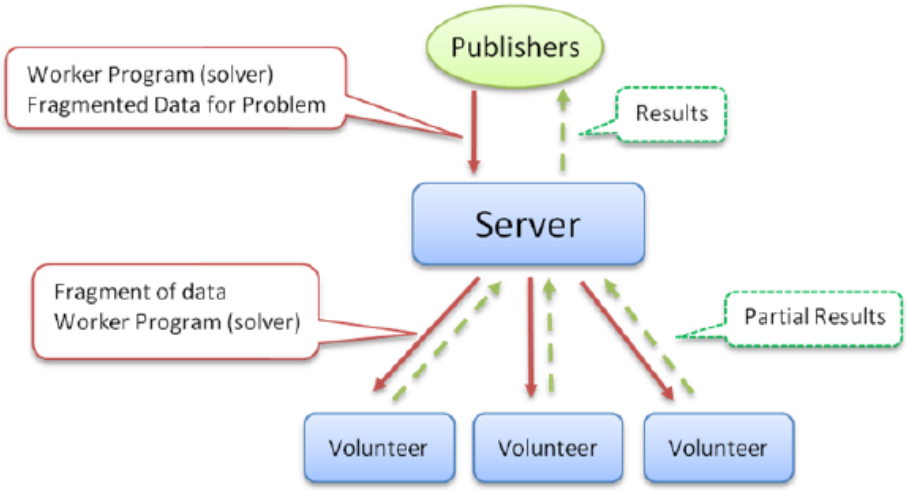
Volunteer computing with BOINC

Joaquim Picó & Ian Palacín

Introduction to Volunteer Computing:

Volunteer computing is a type of distributed computing in wich people donate their computers' unused resources to a research oriented project. This provides substantial processing power to researchers at minimal cost, wich makes research more affordable and efficient. This type of computing is made mostly using cloud distributed systems. In these systems a program is running on a volunteer's computer and periodically contacts the research application to request jobs and report results. The one in charge of the communication between the volunteer program and the researchers' application is the Middleware.

Volunteer Computing Architecture



This picture shows the basic structure of a volunteer computing distributed system. We can see that on the top of the diagram we have the publishers/reserchers, that are the one's that need the computational power to perform some task. They send a fragment of the problem to a middle of the way server, and then the server sends it to a volunteer worker. When the volunteer finishes the work, it returns to the middle of the way server the results and then the server replicates this results to the publishers/researchers.

What is BOINC

BOINC comes from Berkeley Open Infrastructure for Network Computing and is an open-source software that acts as a middleware system for volunteer computing and greed computing. It was originally developed to support the

SETI@home project, which purpose was to analyze radio signals. Later it became generalized as a platform for other distributed applications. As we will see, now the applications reach from mathematics and medicine to climatology and astrophysics. BOINC has been ranked as the largest computing grid in the world. As a volunteer computing platform, the interoperability is crucial to be able to reach all the people, it supports various operating systems, including Windows, MacOS, Android and Linux.

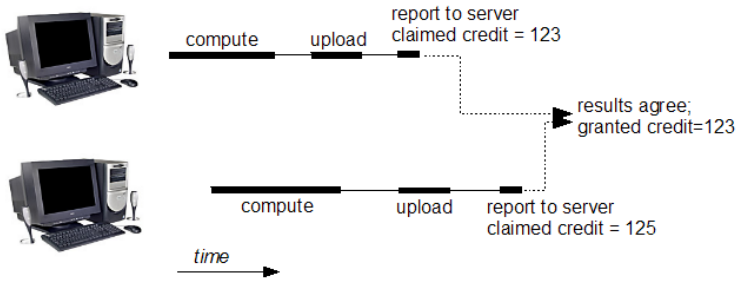
How BOINC works

Client workflow

1. Once a client has BOINC, it starts getting tasks from the project's **scheduling server**. The taks depend on the computation power of the client.
2. The client downloads the executable and the input files from the project's **data server**.
3. The client runs the executables, producing output files.
4. The client uploads the output files tho the **data server**.
5. Later (it can take up to several days), the client reports the completed task to the **scheduling server**, and gets new tasks. This cicle is repeated indefinitely and BOINC does it all.

Credit

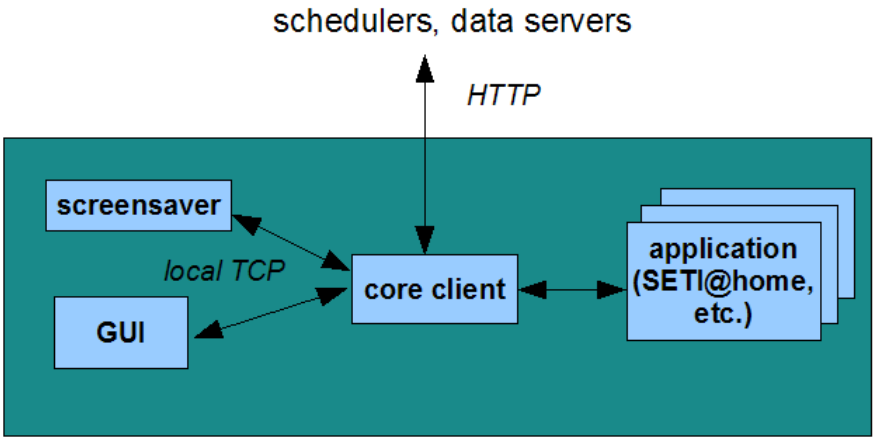
The project's server keeps track of the amount of work a client has done, this is called credit. To ensure that the credit is granted fairly, most BOINC projects work as follow: * Each task is sent to **two or more** computers. * When a computer reports a result, it claims a certain amount of credit, based on how much CPU time was used. * When at least two results have been returned, the server compares them. If the results agree, then users are granted the smaller of the claimed credits.



How the software works

The shcedulers and data servers are installed on computers owned and managed by the project's owner. The programs that are installed on the client computer

are the following: * The **core client** (or client, named boinc.exe) communicates with the external servers via **HTTP** to get and report work. This is the part that runs and control the application. * **Applications** are the programs that do scientific computing. * The **GUI** is the part that provides a graphical interface. It communicates with the core client by a TCP connection. Normally is a local connection. * The **screensaver** runs when the client is away from the computer. It also communicates whith the core client by a TCP connection.



Examples of Volunteer Compting with BOINC

- **PrimeGrid**: A Volunteer Cloud Computing Network dedicated to find new prime numbers.
- **Minecraft@Home**: A Volunteer Cloud Computing Network dedicated to understand the fundamental laws of Minecraft to answer unanswered questions regarding the features and true limits of the game.
- **NumberFields@home**: is a research project that uses Internet-connected computers to do research in number theory.
- **GPUGrid.net**: is a distributed computing infrastructure devoted to biomedical research. Thanks to the contribution of volunteers, GPUGRID scientists can perform molecular simulations to understand the function of proteins in health and disease.
- **Einstein@Home**: uses your computer's idle time to search for weak astrophysical signals from spinning neutron stars (often called pulsars) using data from the LIGO gravitational-wave detectors, the Arecibo radio telescope, and the Fermi gamma-ray satellite.
- **Rosseta@Home**: By running Rosetta@home on your computer you will speed up and extend our efforts to design new proteins and to predict their 3-dimensional shapes. Proteins are the molecular machines and building blocks of life.

Going deeper into an example

Here we will go deep on the Minecraft example, in this case study the purpose was to find a game seed of a random generated world in the search of it starting from an image. We will focus on this project, called the PackPNG project, that tries to find the world seed starting from a photo of a place in the random world. This project is really a huge one, not only with computing complexity, also about mathematics. But we will go straight to the computing part, and where distributed computing is applied. They ended having 2^48 seeds to check, so they thought that running this part in a single computer would be impossible. In order to accelerate the computation they used BOINC to make a distributed volunteer computing system in wich 3500 people ended contributing. After some months and the 95% of the 2^48 seeds checked, they found the seed of the world that they were searching. This part will be further commented as it is the most interesting for us.