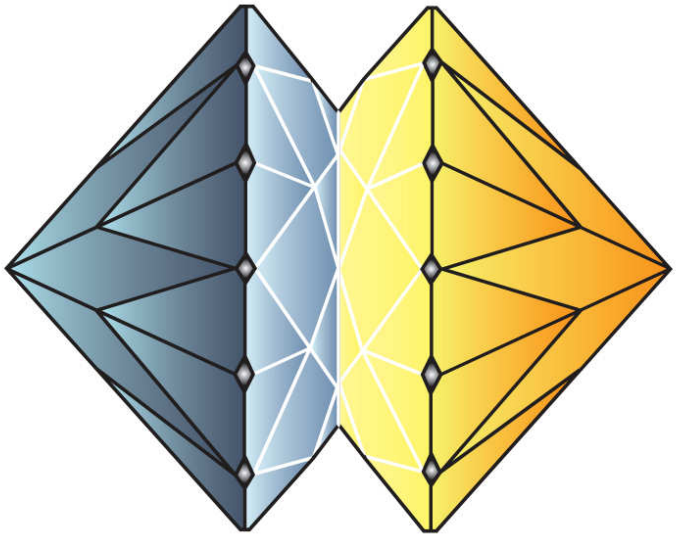


Introduction to BOINC

József Kovács,
smith@sztaki.hu
MTA SZTAKI

The EDGeS project receives Community research funding

Outline

- Volunteer Computing
- BOINC in general
- BOINC main features
- Inside the BOINC server
- Preparing a BOINC server

Volunteer Computing

What is volunteer computing?

- **Volunteer computing** is an arrangement in which people (**volunteers**) provide computing resources to **projects**, which use the resources to do distributed computing and/or storage.
  - Volunteers are typically members of the general public who own Internet-connected PCs. Organizations such as schools and businesses may also volunteer the use of their computers.
  - Projects are typically academic (university-based) and do scientific research.

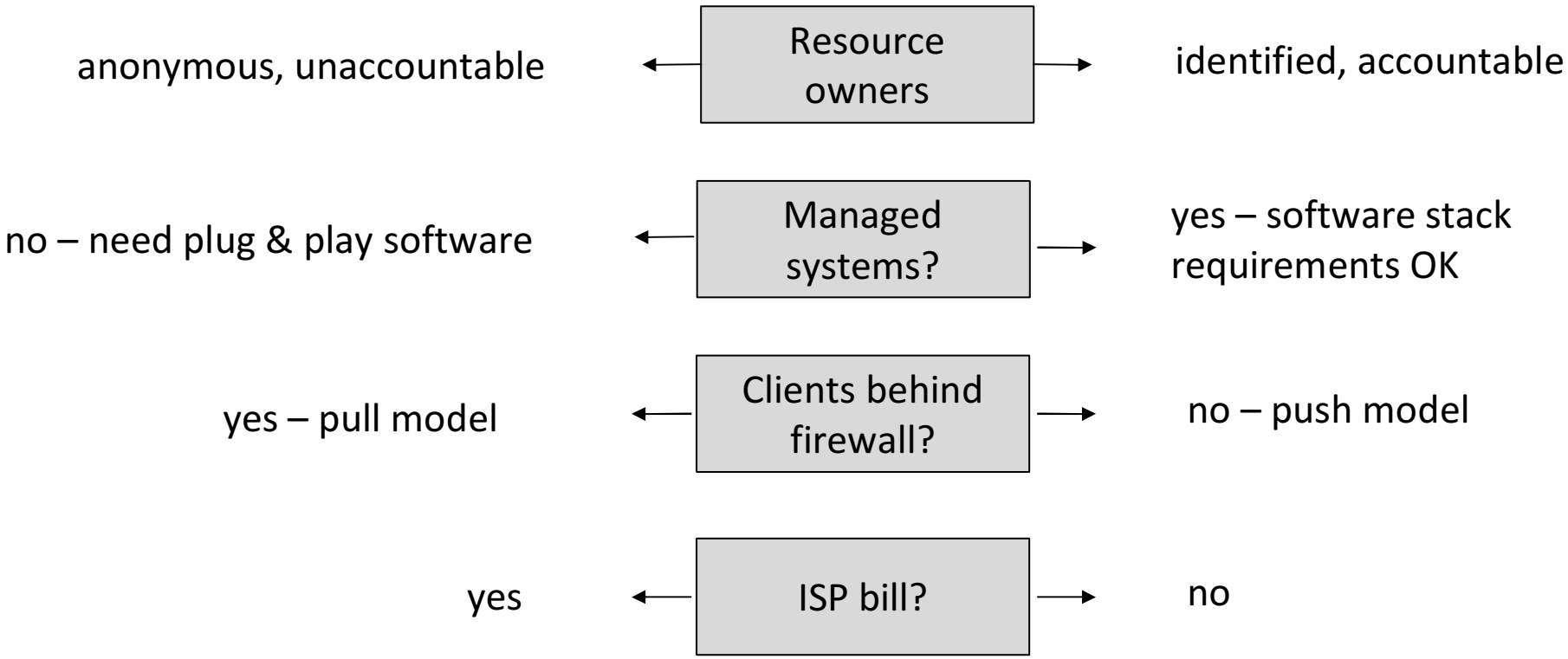
Why is volunteer computing important?

- Because of the huge number (> 1 billion) of PCs in the world, volunteer computing supplies more computing power to science than does any other type of computing. This advantage will increase over time, because consumer products such as PCs and game consoles will advance faster and that there will be more of them.
- Volunteer computing power can't be bought; it must be earned. A research project that has limited funding but large public appeal can get huge computing power. In contrast, traditional supercomputers are extremely expensive, and are available only for applications that can afford them (for example, nuclear weapon design and espionage).

PCs vs. Supercomputers

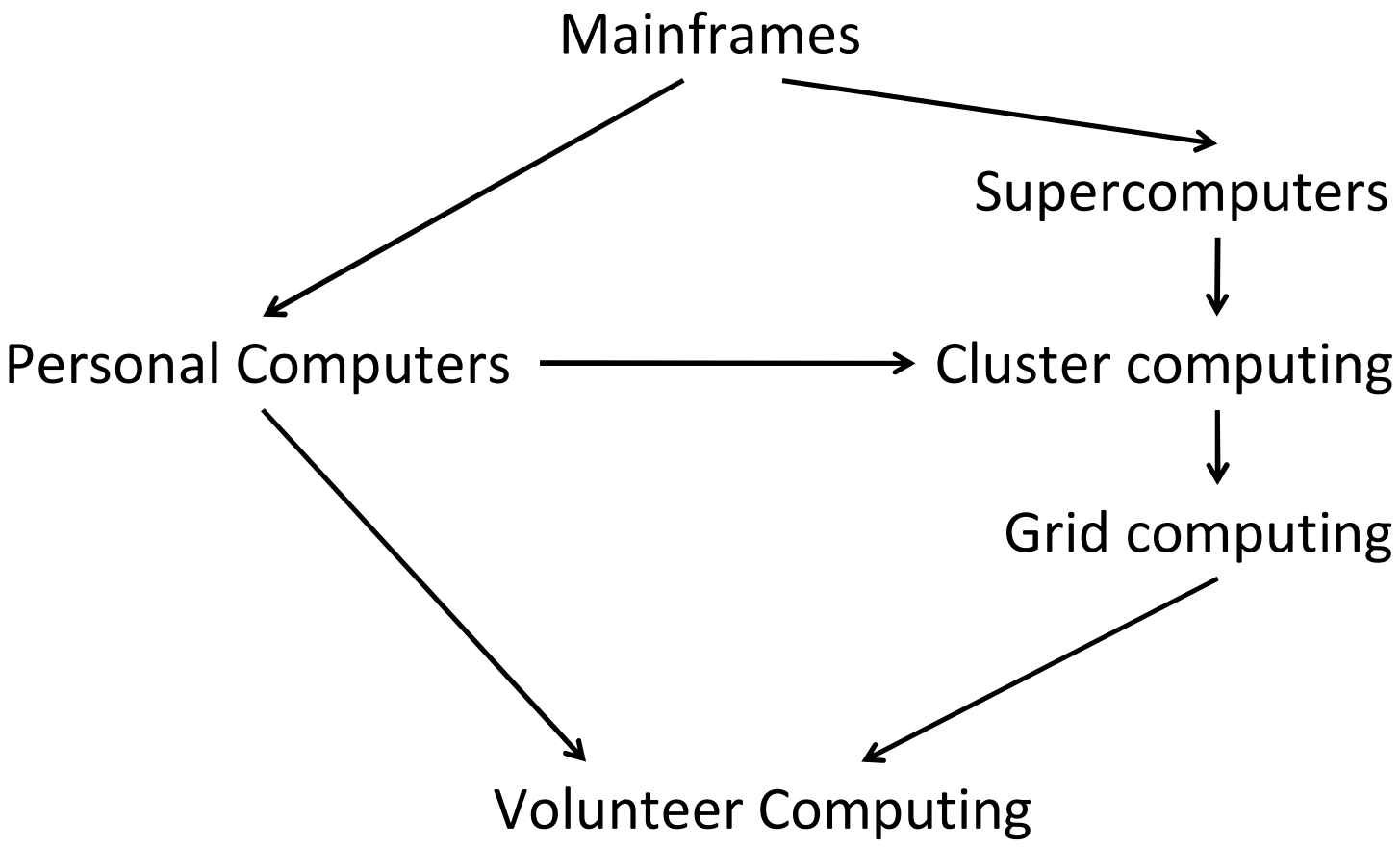
- PCs trail supercomputers by about 20 years
- 100,000 PCs == 1 supercomputer
- There are ~1 billion PCs on the Internet
- Consumer products (PCs, game consoles) are getting faster
- PC owner buys PC, maintains it, buys electricity
- But: PCs are unreliable and untrusted

Volunteer computing != Grid computing



Volunteer computing is not “peer-to-peer computing”

Evolution of Paradigms



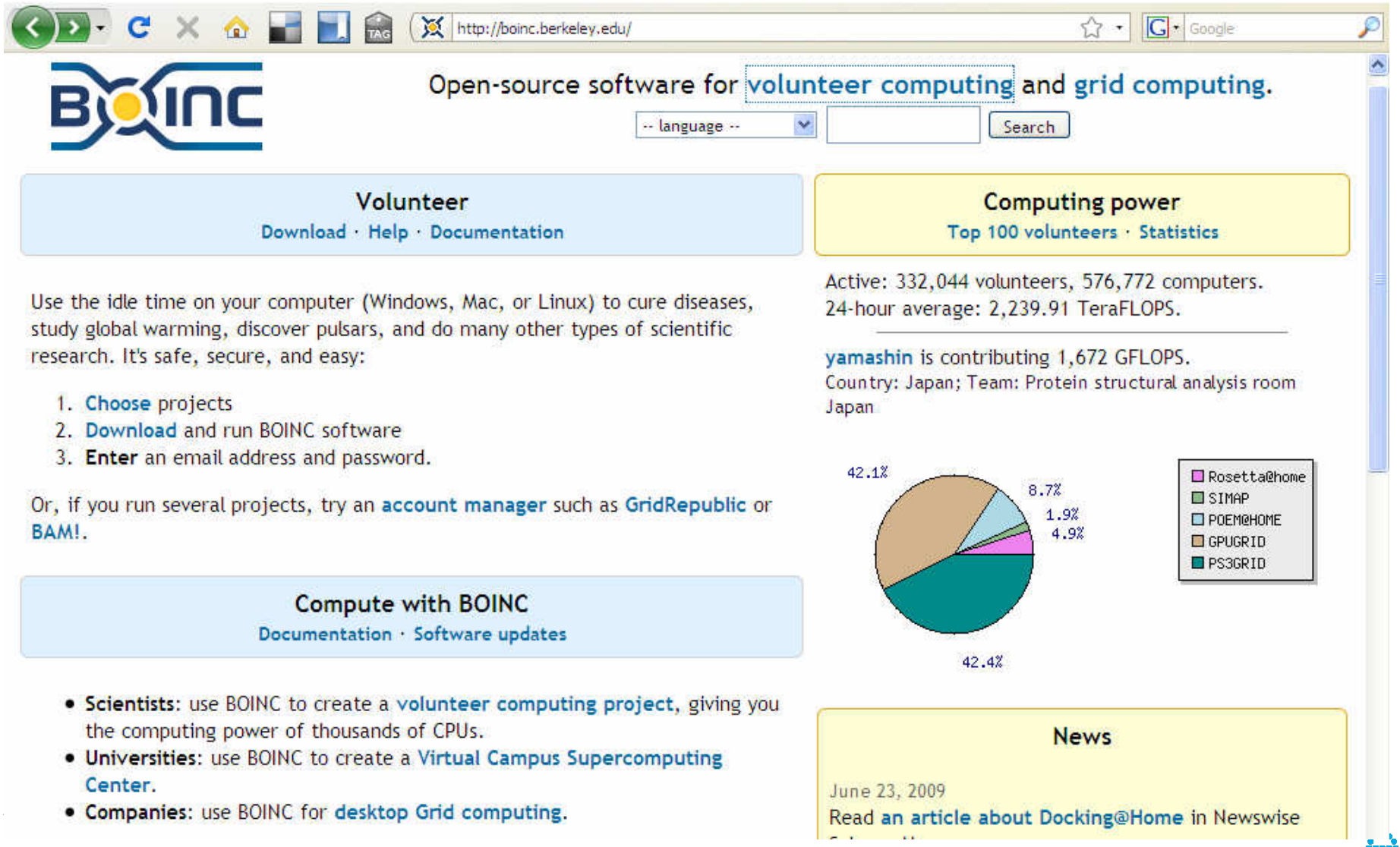
BOINC in general

Berkeley Open Infrastructure for Network Computing

BOINC

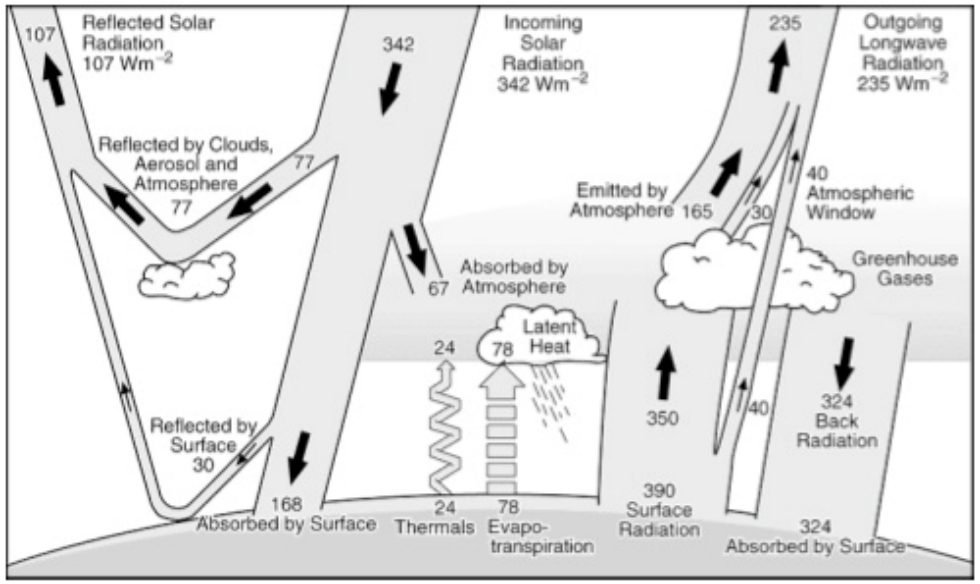
- Middleware for volunteer computing
- Open-source (LGPL)
- Application-driven
- Goals
  - lots of independent projects
  - support for diverse applications
  - client participation in multiple projects

BOINC homepage

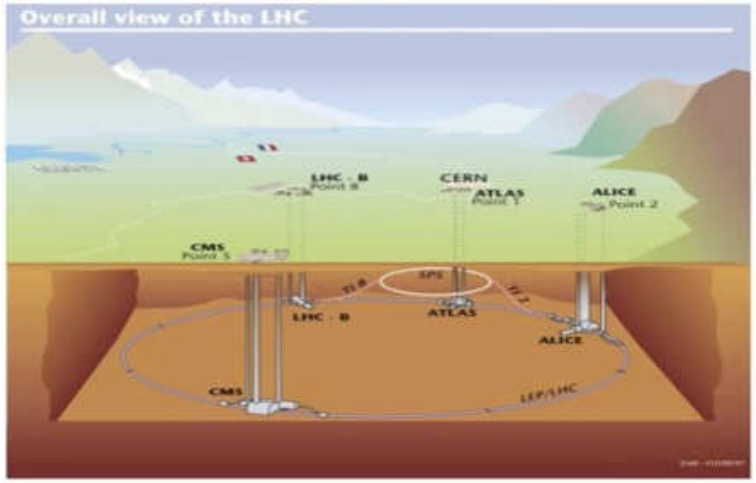


Some BOINC Projects

Climateprediction.net
(Oxford University)



LHC@home
(CERN)



EDGeS

SETI@home (U.C. Berkeley)

SZTAKI Desktop Grid (Hungary)

SZTAKI DESKTOP GRID

BOINC main features

Main features of BOINC

- **Project autonomy**
  - Many projects use BOINC. Projects are independent; each one operates its own servers and databases.
- **Volunteer flexibility**
  - Volunteers can participate in multiple projects; they control which projects they participate in, and how their resources are divided.
- **Flexible application framework**
  - Existing applications in common languages (C, C++, Fortran) can run as BOINC applications with little or no modification.
- **Security**
  - BOINC protects against several types of attacks. For example, digital signatures based on public-key encryption, etc.
- **Server performance and scalability**
  - The BOINC server software is extremely efficient, and highly scalabe, so that a single mid-range server can dispatch and handle millions of jobs per day.
- **Source code availability**
  - BOINC is distributed under the Lesser General Public License. However, BOINC applications need not be open source.
- **Support for large data**
  - BOINC supports applications that produce or consume large amounts of data, or that use large amounts of memory. Data distribution and collection can be spread across many servers. Users can specify limits on disk usage and network bandwidth.
- **Multiple participant platforms**
  - The BOINC core client is available for most common platforms (Mac OS X, Windows, Linux and other Unix systems).
- **Open, extensible software architecture**
  - BOINC makes it possible for third-party developers to extend BOINC.
- **Volunteer community features**
  - BOINC provides web-based tools, such as message boards, user profiles, and private messaging, that encourage volunteers to form online communities.

Which application?

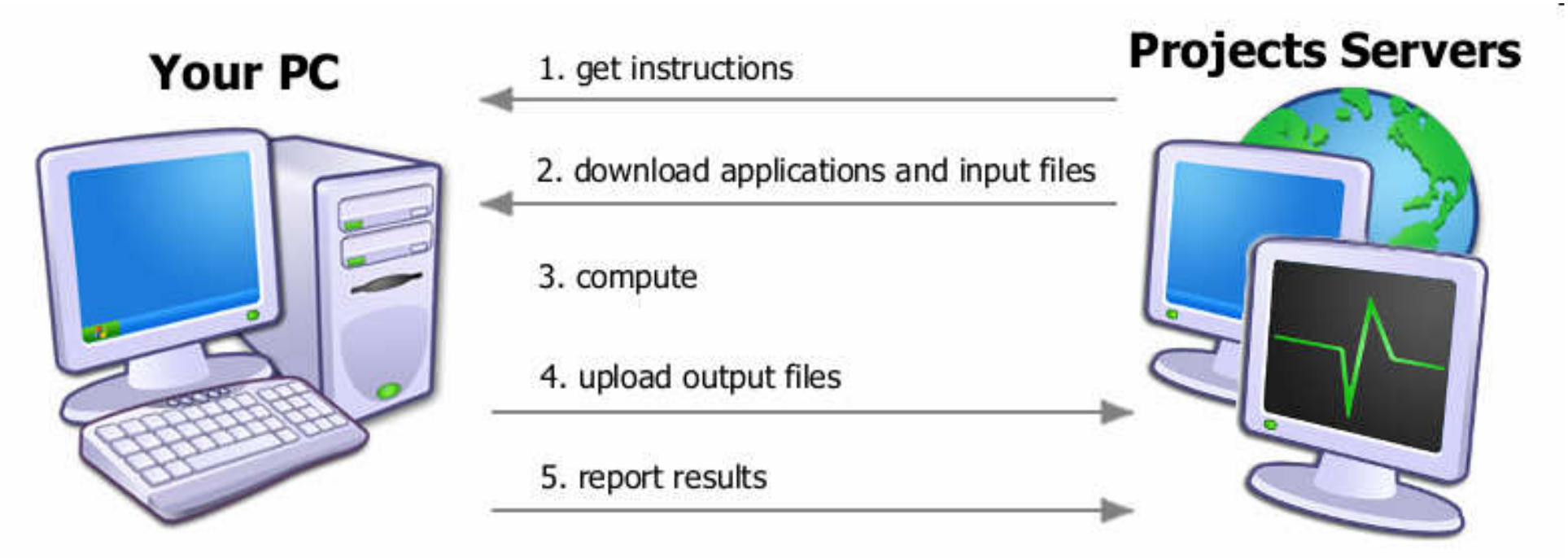
- BOINC is designed to support applications that have
  - large computation requirements
  - large storage requirements
  - or both.

  The main requirement of the application is that it be divisible into a large number (thousands or millions) of jobs that can be done independently.
- In case of volunteer resources, there are additional requirements:
- **Public appeal**
  - An application must be viewed as interesting and worthwhile by the public in order to gain large numbers of participants. A project must have the resources and commitment to maintain this interest, typically by creating a compelling web site and by generating interesting graphics in the application.
- **Low data/compute ratio**
  - Input and output data are sent through commercial Internet connections, which may be expensive and/or slow. As a rule of thumb, if your application produces or consumes more than a gigabyte of data per day of CPU time, then it may be cheaper to use in-house cluster computing rather than volunteer computing.

Key expressions

- **Project**
  - A **project** is an entity that does distributed computing using BOINC. Projects are independent.
- **Application**
  - An **application** includes several programs (for different platforms) and a set of workunits and results. A project can include multiple applications.
- **Platform**
  - A **platform** is a compilation target - typically a combination of a CPU architecture and an operating system. BOINC defines a set of standard platforms.
- **Application versions**
  - An application program may go through a sequence of versions. A particular version, compiled for a particular platform, is called an **application version**. An application version consists of one or more files.
- **Workunit**
  - A **workunit** is a computation to be performed, i.e. a "job". It may include any number of input files. It has various attributes, such as resource requirements and deadline.
  - A workunit is associated with an application, not with an application version. In other words, you don't specify what platform the job is to be run on.
- **Result**
  - A **result** describes an instance of a computation, either unstarted, in progress, or completed. Each result is associated with a workunit. In some cases there may be several instances, or "replicas", of a given workunit.
- **Account**
  - Each volunteer in a project has an **account**, identified by an email address and password. An account has an associated amount of **credit**, a numerical measure of the work done by that volunteer's computers.
- **Credit**
  - The project's server keeps track of how much work your computer has done.
- **Clients and attachment**
  - Volunteers run a program called the **BOINC client**; this is the only software they manually download.
  - A given computer running the BOINC client can be **attached** to accounts on one or more project. Each attachment has a **resource share**. If a computer is attached to multiple projects, its resources are divided among them in proportion to their resource shares.

How BOINC works?



How to participate

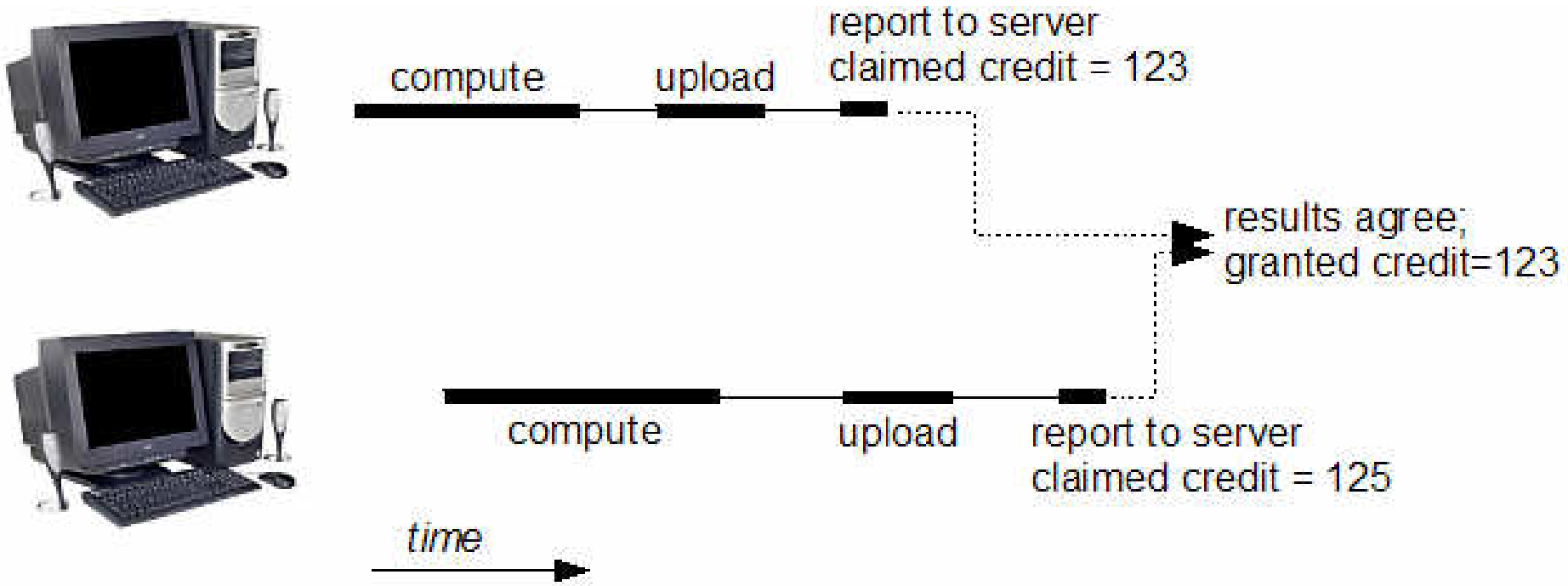
- Install and run BOINC client software
  - http://boinc.berkeley.edu
  - Available for Windows, Mac OS X, Linux
- Enter the URL of a project
  - e.g.: http://szdg.lpds.sztaki.hu/szdg
- Enter your email address and password
- Done!

BOINC Features

- Credit system
  - Volunteer competition
- Community features
  - Message boards
    - Science, technical, social
  - User profiles
  - Teams
  - Translations
  - Web sites

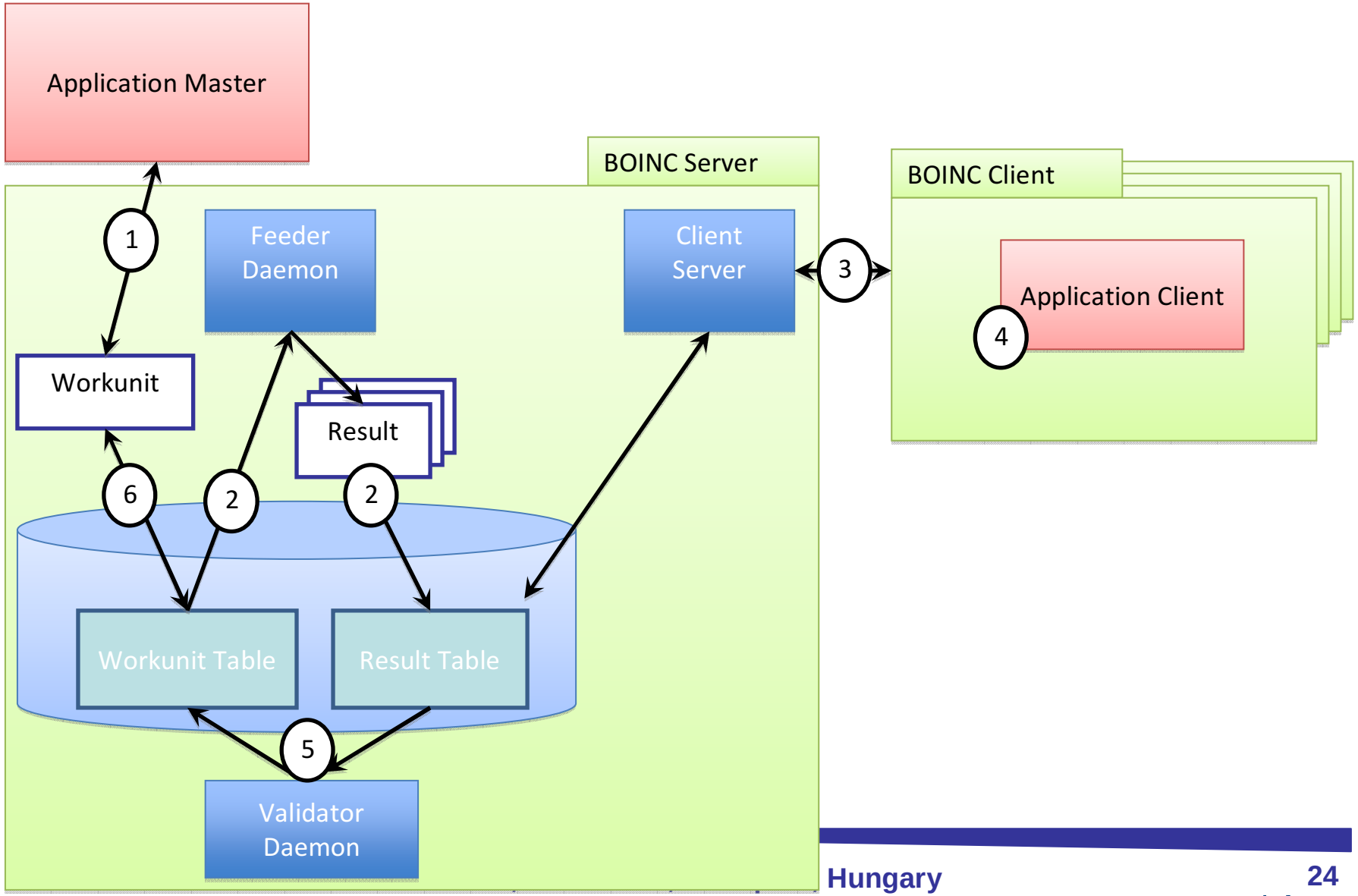
BOINC Credit

- The project's server keeps track of how much work your computer has done; this is called **credit**. To ensure that credit is granted fairly, most BOINC projects work as follows:
  - Each task may be sent to two 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.

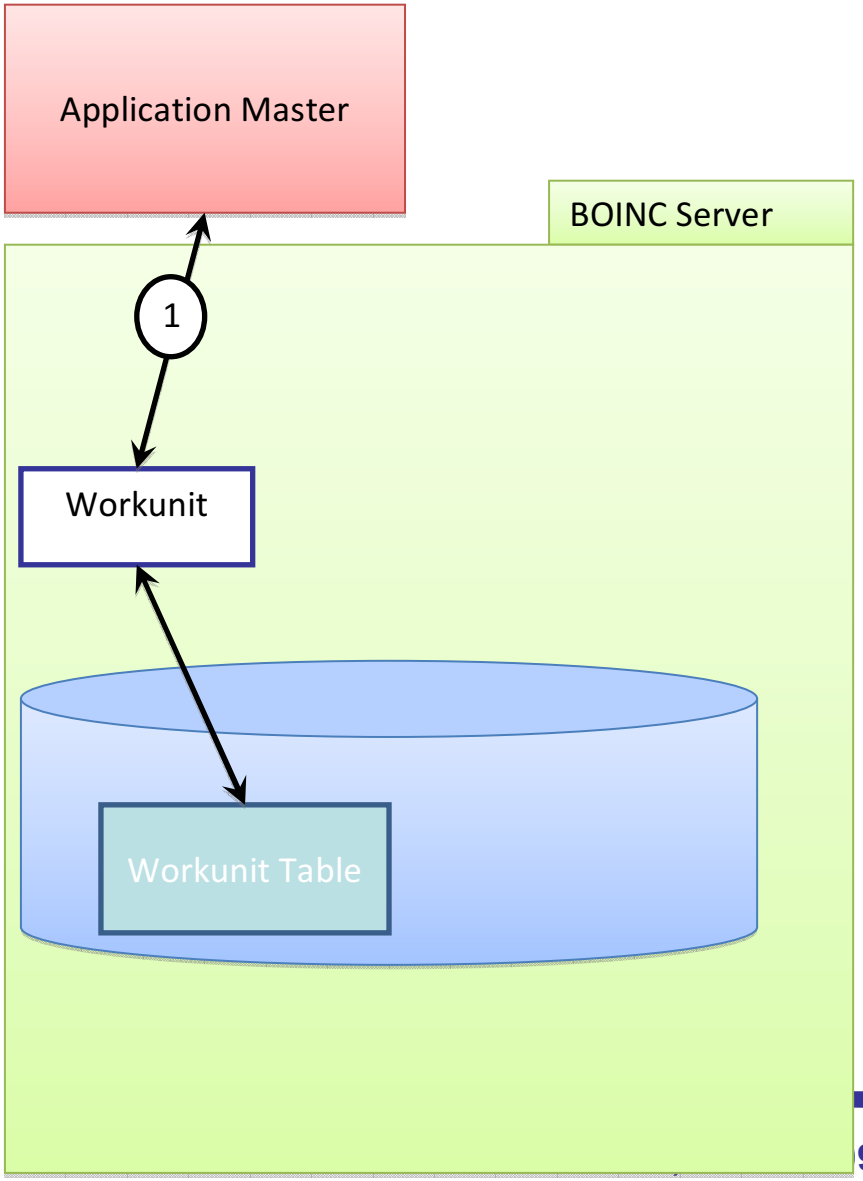


Inside the BOINC server

Under the hood

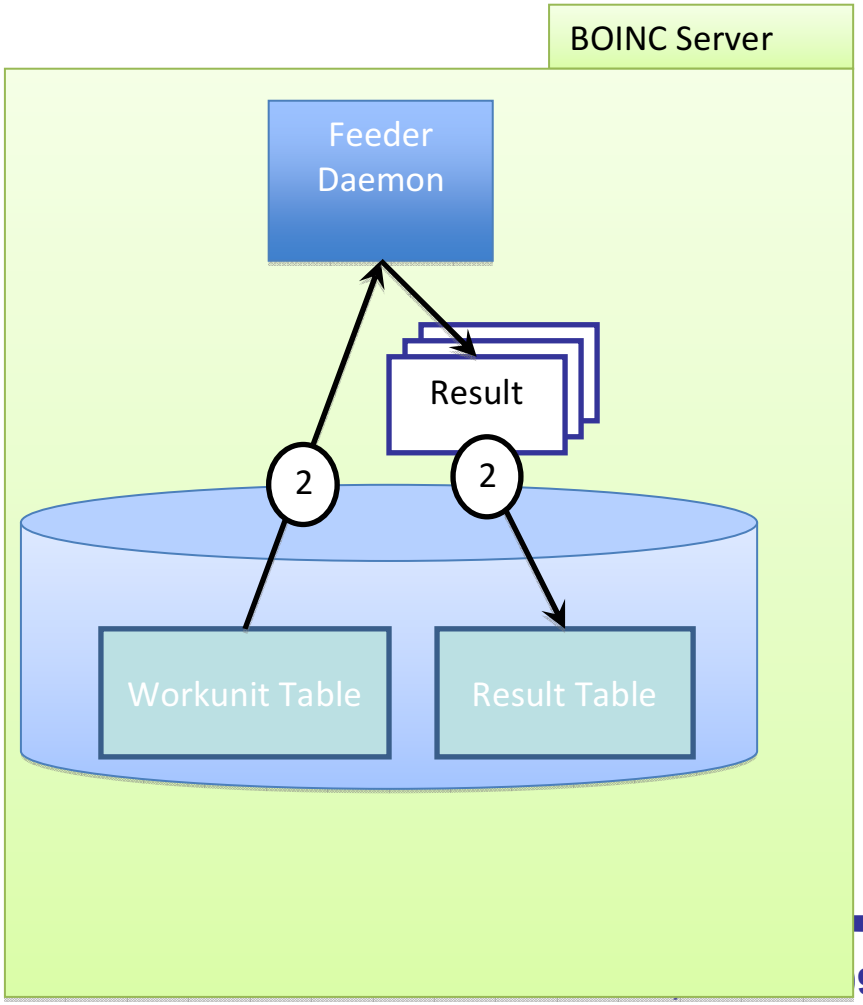


Under the hood



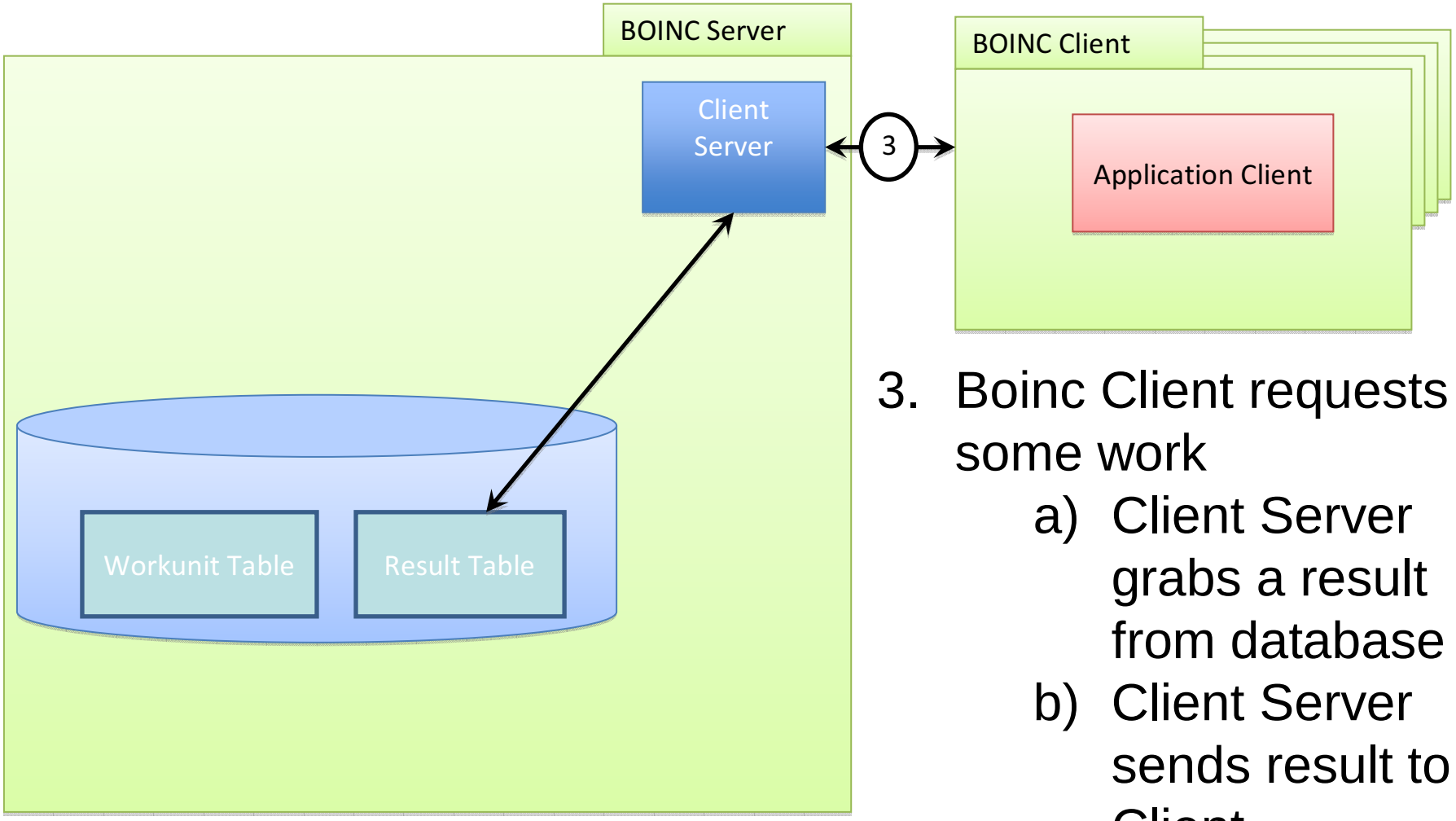
1. Application master submits workunit(s)

Under the hood



2. Feeder daemon checks for new workunits
   a) Creates Results for redundant computing
   b) Registers Results in the result table of the database

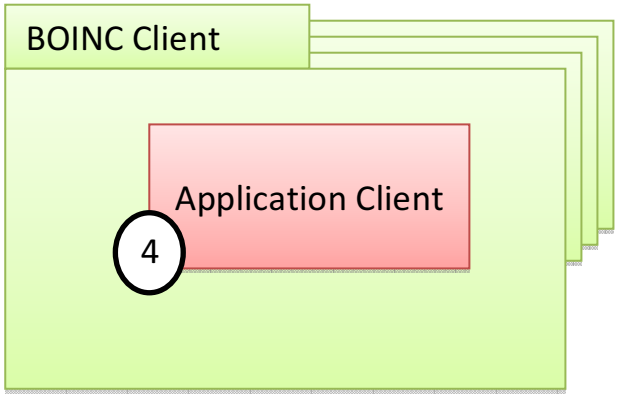
Under the hood



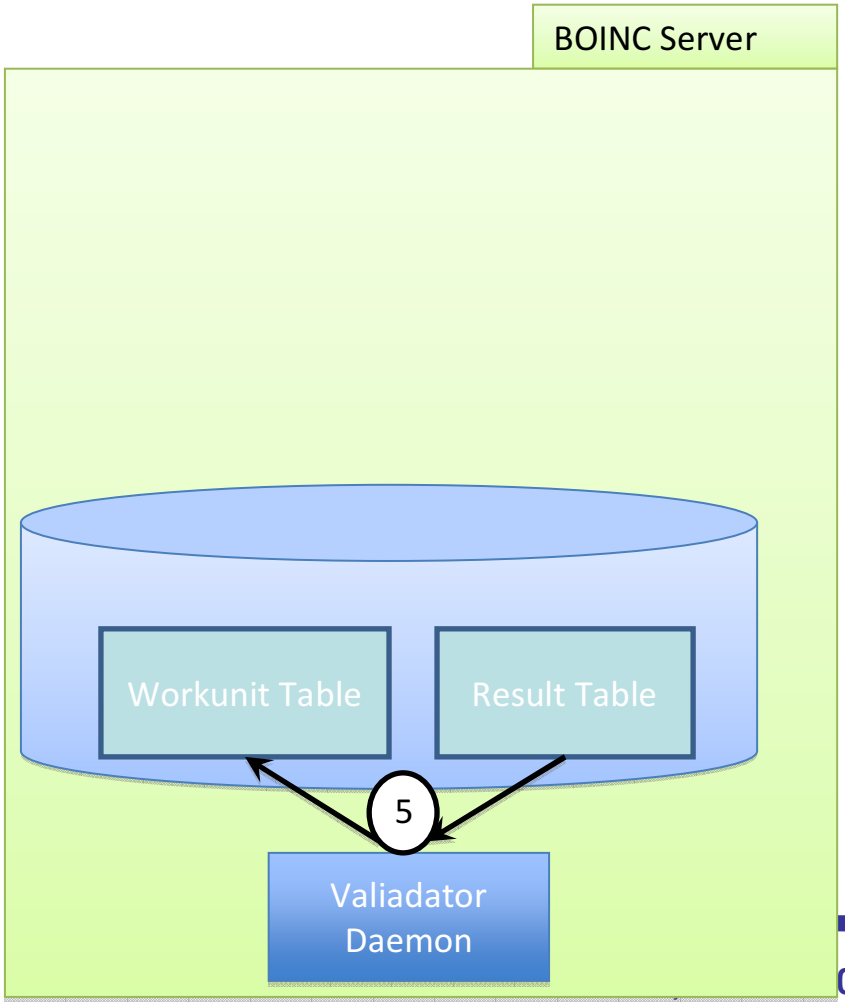
3. Boinc Client requests some work
    a) Client Server grabs a result from database
    b) Client Server sends result to Client

Under the hood

4. BOINC Client processes result
   a) Downloads application client if it does not have it already
   b) Executes client application with the input found in the downloaded result
   c) Upon finish the output of the result is sent to the BOINC Server

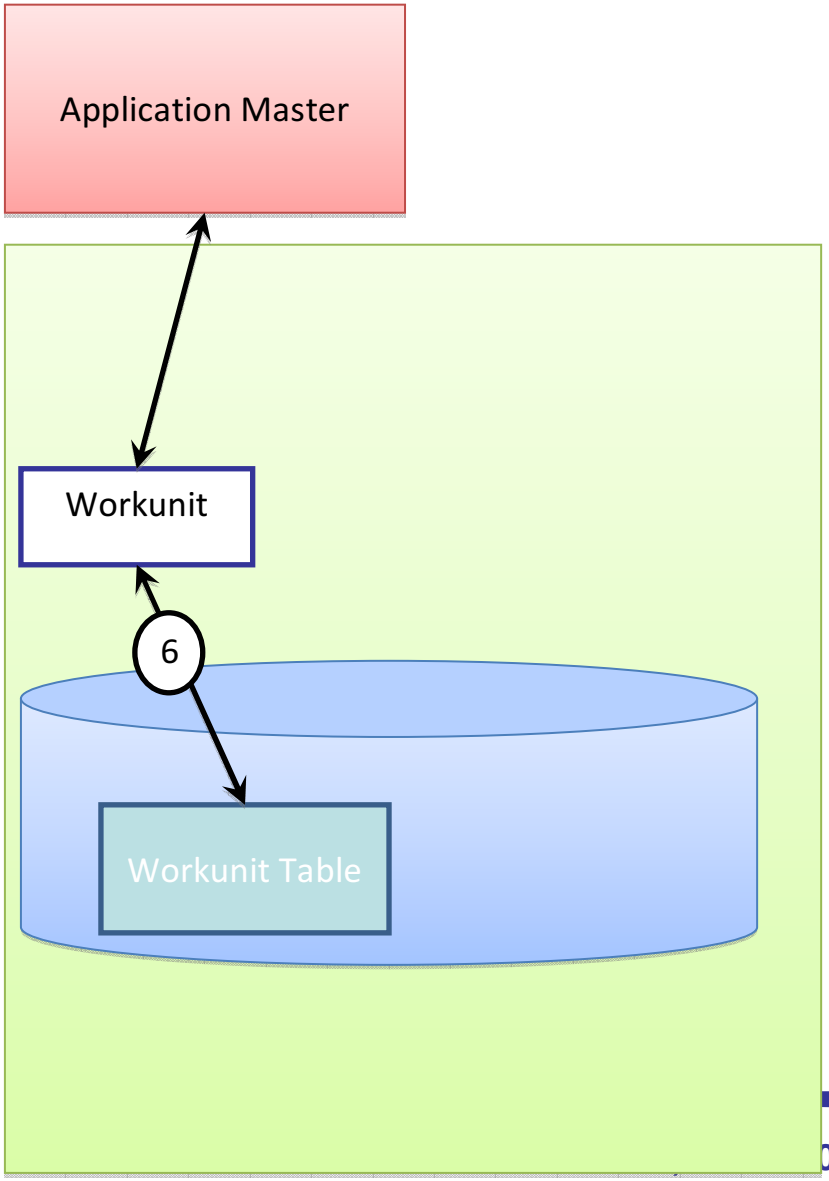


Under the hood



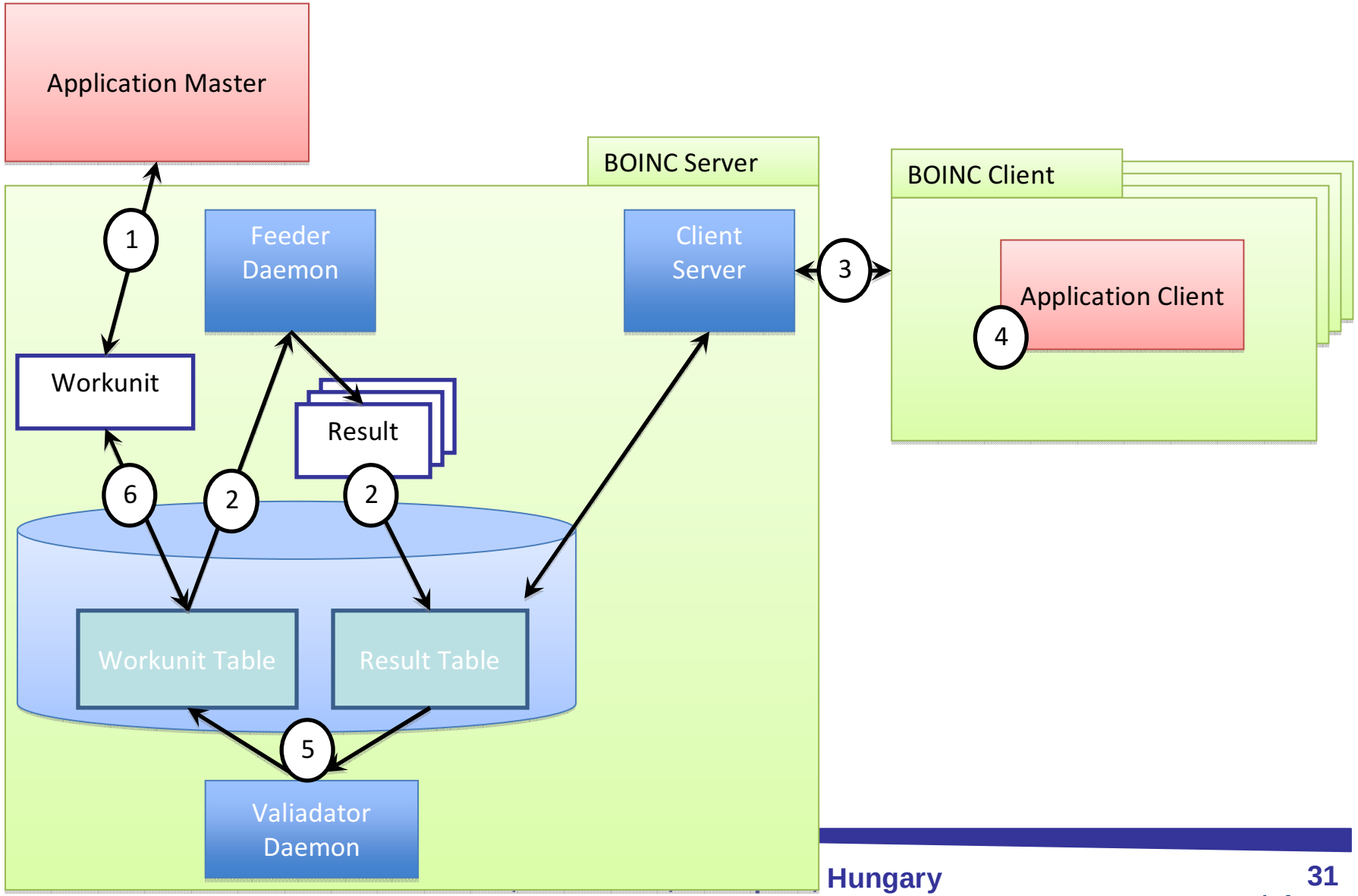
5. Validator Daemon checks for finished results
   a) If all of the results of a workunit has been finished they get compared by the validator
   b) If there's a majority match:
      i. Output is passed to the workunit
      ii. Results are marked as deletable
      iii. State of the workunit is set to Finished
   c) If there's no match:
      i. New results are generated and registered in the result table
      ii. If there's still no consensus after a certain number of results, the workunit is considered to be a failure

Under the hood



6. Application Master polls the workunit table for any workunit in finished state
   a) If found the output of the workunit is grabbed and processed

Under the hood



Preparing a BOINC server

Preparing a BOINC Server

- Ingredients:
  - 1 piece of hardware
  - BOINC server package
  - Application

Preparing a BOINC Server

- 1 pcs of hardware
  - Capable of running Debian linux
  - CPU with at least 1 GHz
  - Min. 512 MB of memory (the more the better)
  - Fast internet connection (upload needs to be fast as well)
  - Plenty of available storage (depends on application)

Preparing a BOINC Server

- BOINC server package
  - Available at http://www.desktopgrid.hu
  - Installation guide is also available there

Preparing a BOINC Server

- Application
  - Creating Application Master
  - Creating Application Client
    - Compiling for multiple platforms
  - Creating Application Validator
  - Registering Application on the BOINC Server

Preparing a BOINC Server

- Application
  - Creating Application Master

The job of the Master is to create workunits

- Usually by breaking a big input into smaller pieces
- Submit the workunits in the BOINC server

Development is aided by DC-API

- Available at http://www.desktopgrid.hu
- Examples and documentation also available

Preparing a BOINC Server

- Application
  - Creating Application Client

The job of the client is to process the input found in the results downloaded from the BOINC Server

  - Usually a non-boinc application exists, that can be modified with DC-API to support execution in a BOINC environment
  - DC-API aids development
  - Application is executed in a “rough” environment on the client side: should support checkpointing to be able to survive restarts and other conditions

Preparing a BOINC Server

- Application
  - Creating Application Client
    - Compiling for multiple platforms

The majority of BOINC clients use the Windows platform

Preparing a BOINC Server

- Application
  - Creating Application Validator

Output calculated by clients is not trustworthy

- Computation errors
- Malicious user intervention

Redundancy is used to overcome this condition

- Applicatin specific validator is needed that compares several outputs belonging to the same workunit and decides if they can be accepted or refused

Preparing a BOINC Server

- Application
  - Registering Application on the BOINC Server
    - Tutorial is available at http://www.desktopgrid.hu
  - Copy client app executable to the server
  - Register client apps with command-line utility
  - Copy master app executable to the server
  - Add master app to the xml listing server daemons
  - Add application to the xml listing registered applications
  - Register application with command-line utility

Conclusion

- Scientific computing
  - Will always need more computing power
- Volunteer computing
  - The computing paradigm of the future
- BOINC
  - Middleware for volunteer computing

Thank you for your attention!

For more information, visit:

http://boinc.berkeley.edu

or

http://www.desktopgrid.hu

If you have any question, use:

http://boinc.berkeley.edu/email_lists.php

or

desktopgrid@lpds.sztaki.hu

Special thanks to
David Anderson and Adam Kornafeld
for the slides!