Custom execution environments in the BOINC middleware

Diogo Ferreira* Filipe Araujo* Patricio Domingues+

CISUC, Dept. of Informatics Engineering, University of Coimbra, Portugal*

Research Center for Informatics and Communications, School of Technology and Management, Polytechnic Institute of Leiria, Portugal+

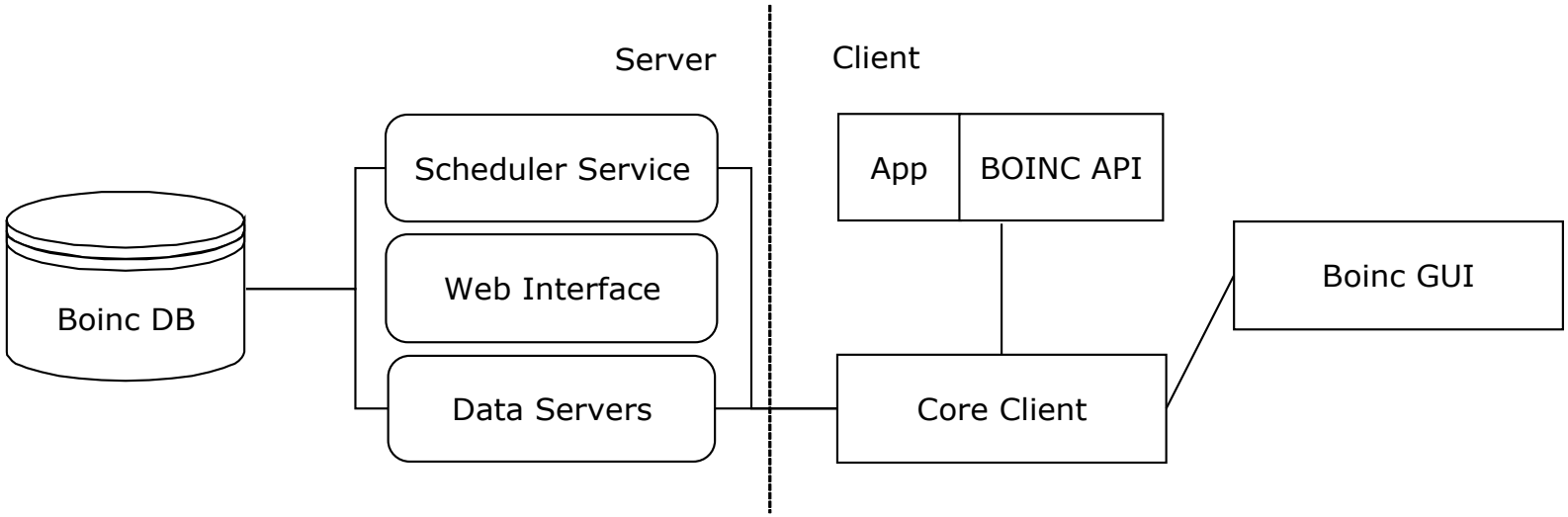
May 26, 2010

Introduction

BOINC is a grid middleware for volunteer machines.
An execution environment is one where work can be executed.
`Libboincexec` is the glue between BOINC and an execution environment.

BOINC

BOINC is a system that powers some of the most powerful volunteer grids.



The core client forks itself and runs the application in the volunteer computer.

Motivation

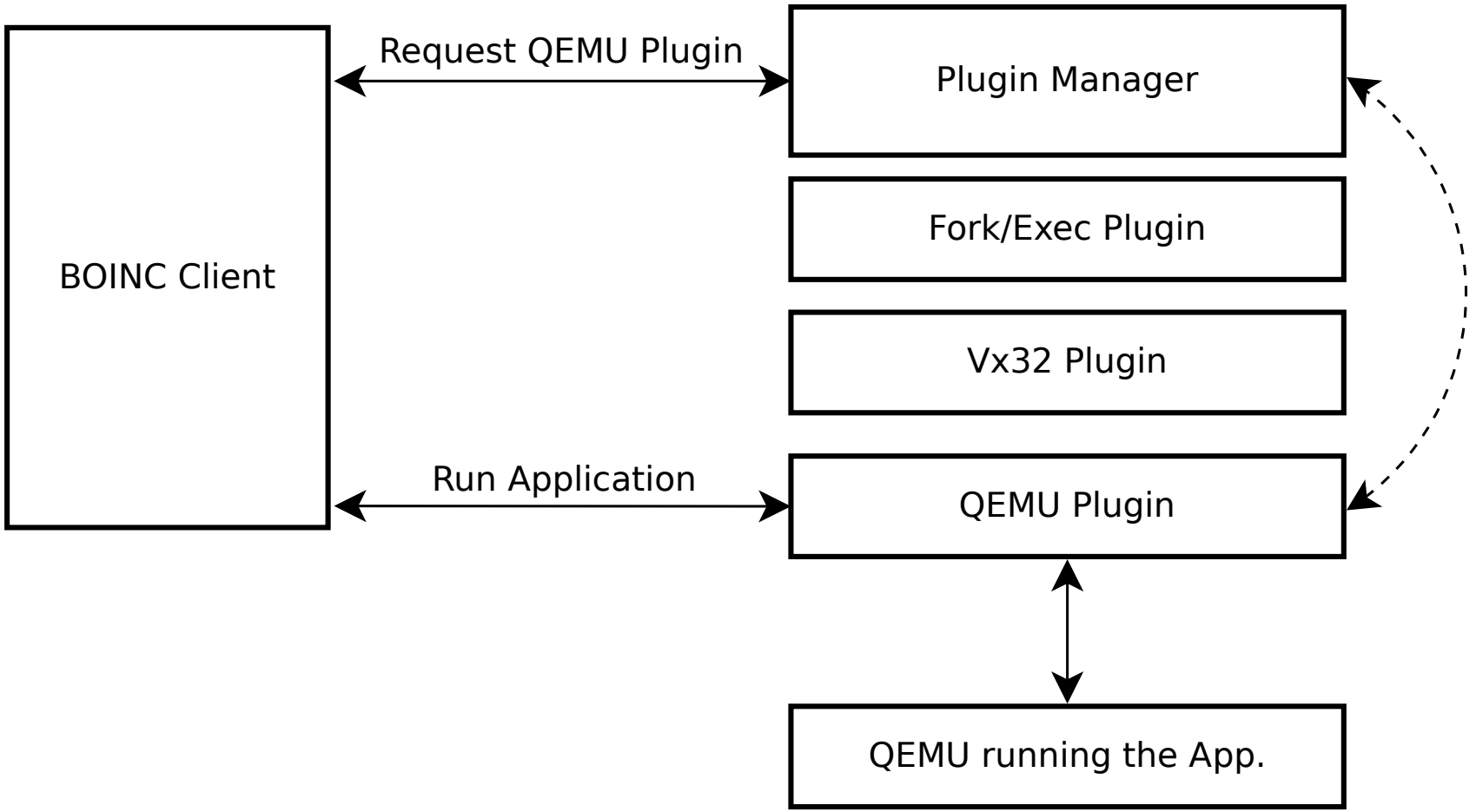
BOINC has some limitations:

- Security issues
- Development time
- Limited to native applications (ie. no Matlab, Java, etc)

`Libboincexec` is an opportunity to tackle these problems.

Libboincexec: Overview

There are two main components: the **Plugin Manager** and the **Execution Plugins**.



Libboincexec: Architecture

The Plugin Manager provides the glue between BOINC and the execution environments.
Since loading libraries is a complex matter we depend on *libltdl*.
The main functions of Plugin Manager are:

- Finding compatible execution plugins
- Expose their functions to BOINC.

Libboincexec: Architecture (II)

Plugins implement a set of abstract functions composed of:

Metadata Contains plugin information, description, possible configuration options.

Prepare work Prepares the execution environment for execution (ie. starting a virtual machine).

File copy Copies files back and forth between host and execution environment.

Execute Runs a command on the execution environment.

Checkpoint Checkpoints the execution environment and the underlying running application.

Libboincexec: Integration

The core client and applications communicate using the BOINC API.
BOINC uses system calls in order to execute the application.
Integration with BOINC can be done in two different levels:

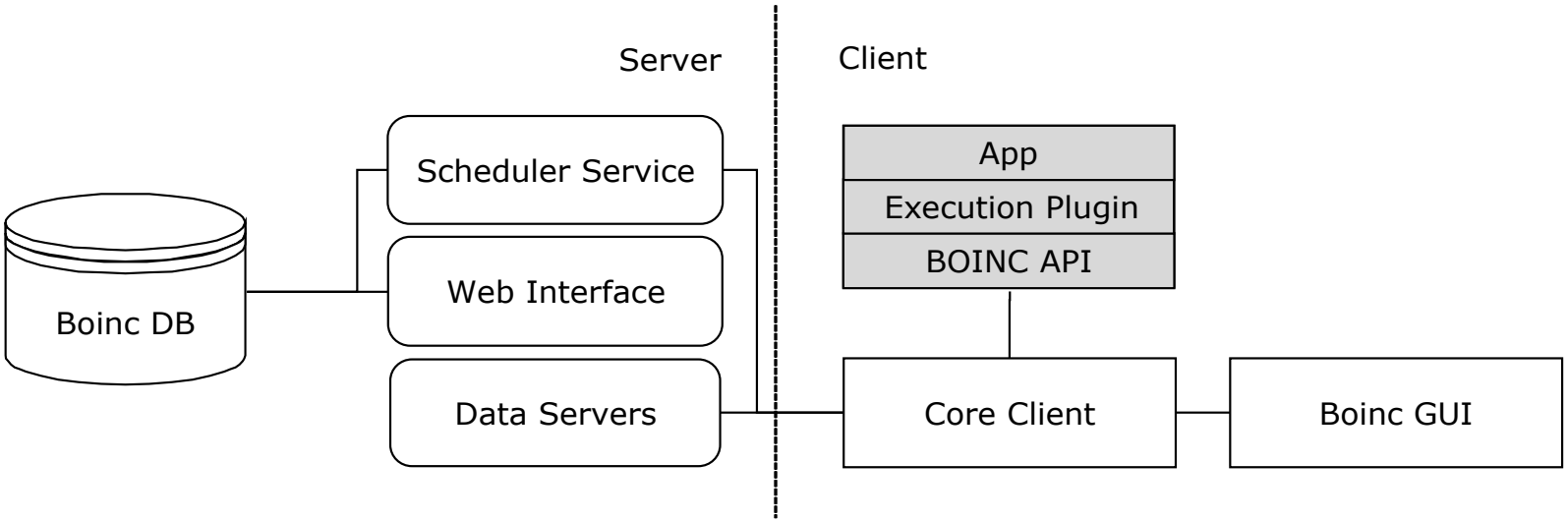
- Wrapper: Let an application implement the BOINC API and translate it to `Libboincexec` calls.
- Execution modification: Let `Libboincexec` replace the execution path in the core client.

Libboincexec: Wrapper (I)

The wrapper communicates with BOINC and runs an *inner application* that performs the actual work.
Integration using a wrapper is done by replacing its life-cycle with Libboincexec.

Libboincexec: Wrapper (II)

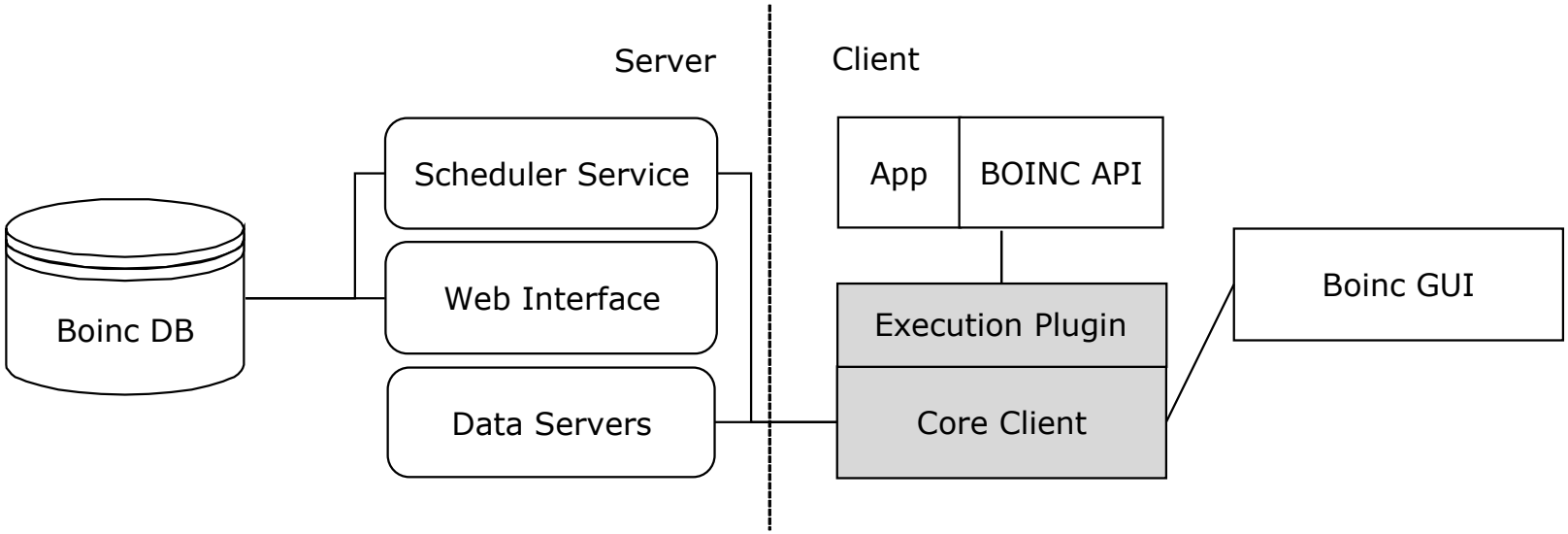
We can see (in gray), the areas that need to be changed in order to integrate using this approach.



Since there are no changes to the core client, this integration is less complex.

Libboincexec: Execution modification

With this method BOINC uses Libboincexec transparently for all jobs. We can see (in gray), the areas that need to be changed in order to integrate using this approach.



Our goal is to keep the client backwards compatible in the process.

Using Libboincexec

There are plenty of use cases for Libboincexec:

- Applications that are hard to port.
- If you need to guarantee the well-being of volunteer machines.
- Applications that require exotic setups.

Using Libboincexec (II)

There is no need to implement the BOINC API.
Using Libboincexec:

- Developers write an application that targets a specific OS.
- A simple XML file is used to describe the input and output files.
- Distribute through BOINC using our wrapper.
- You can tweak the virtual machine image to include the software you require.

Developing BOINC applications becomes easy and secure.

Enhancing BOINC and other possibilities

Out main focus was virtualization, however, `Libboincexec` allows for more:

- An SSH plugin that redirects jobs to a different machine.
- Plugins that redirect jobs to different grid systems.
- Usage without without BOINC.

Open Issues

We currently have a pre-production version of `Libboincexec`, tested in a local desktop grid and are preparing for broader testing. There are however, some open issues:

- Deployment of the virtualization tool is hard to get right.
- Disk space and bandwidth concerns when distributing the image.
- Plugins must handle portability by themselves, we only have Linux and Windows versions at the moment.