Porting Applications to the EDGeS Infrastructure

A comparison of the available methods, APIs, and technologies

Attila Csaba Marosi
MTA SZTAKI

The EDGeS project receives Community research funding

Porting Applications to BOINC

- Porting applications to BOINC is not a straightforward process
- The infrastructure requires the implementation of various components (work unit generator, assimilator, worker application, etc.)
- Developers can find a wide variety of APIs and methods to port their applications
- It is not easy to select the best one for our application
- This presentation gives an overview of the available methods, APIs, and technologies

Application Types

- There are two types of applications:
  - Applications whose source code is available and can be modified easily
  - Legacy applications that cannot be modified (e.g. the source code is not available) or it is not advisable to modify them

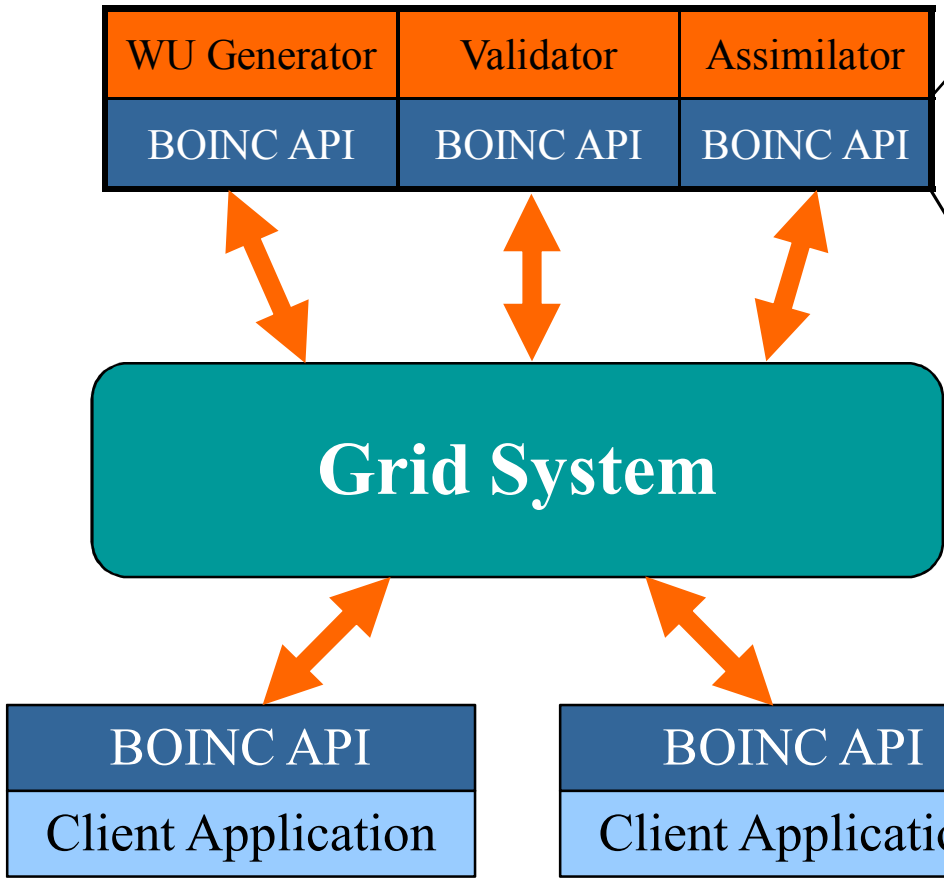
Available Methods, APIs, and Technologies

- Server side
  - BOINC API
  - DC-API
  - gUSE
- Client side
  - BOINC API
  - DC-API
  - Boinc Wrapper technology
  - GenWrapper technology

BOINC API

- The original API, can only be used with BOINC
- A set of C and C++ functions
- Rich feature set at the expense of complexity
- 4 separate applications (WU generator, validator, assimilator, worker)
- Complex porting process

BOINC API Functionality



WU Generator:
- work unit generation, submission
- input and output file handling

Validator:
- validator frameworks (simple, low-level)

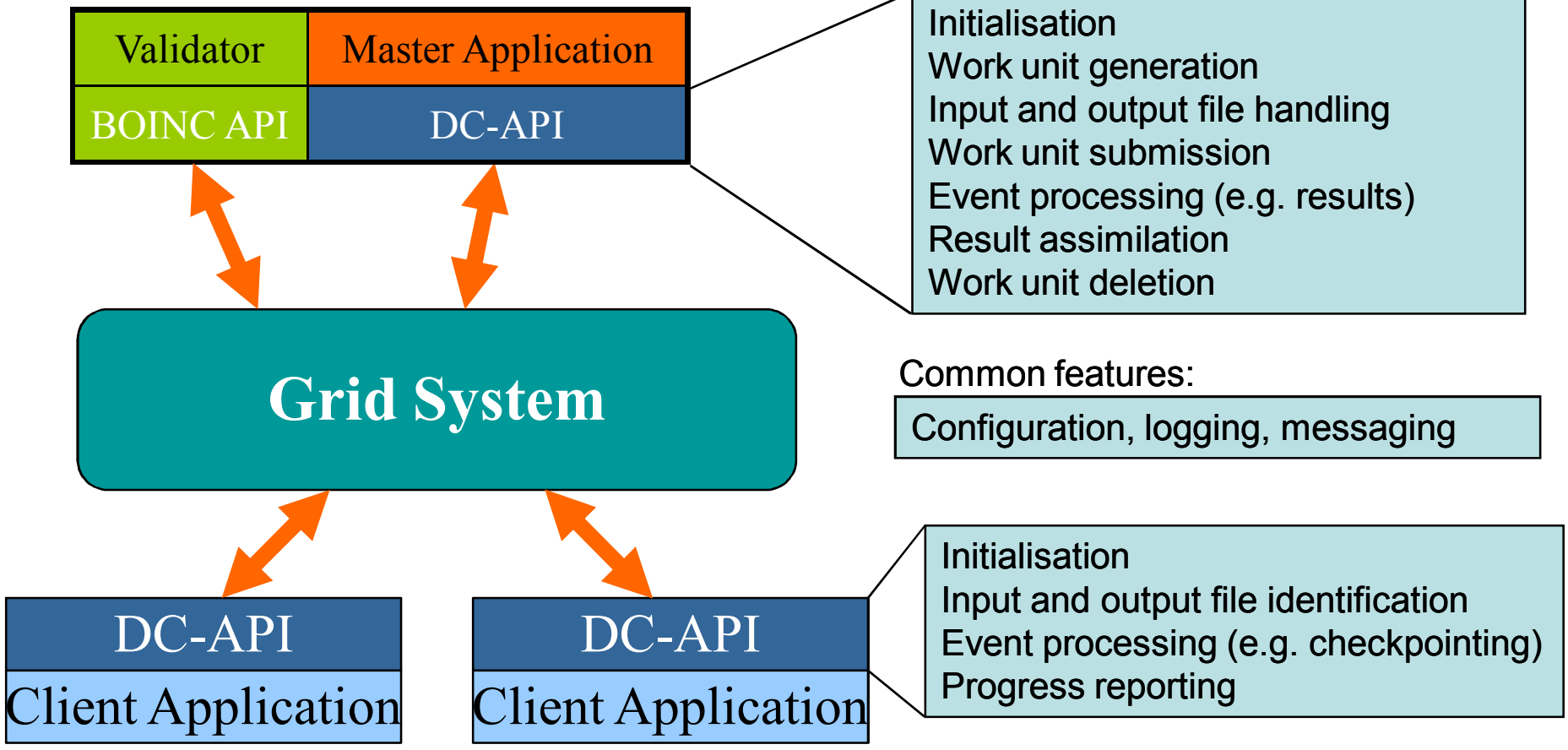
Assimilator:
- handler function linked with a special program

Initialisation
Input and output file identification
I/O wrappers
Checkpointing
Progress, credit reporting
Critical sections
Atomic file update
Time handlers
Graphics API (screen savers, icons)
Other APIs (status, diagnostics, etc.)

DC-API

- Distributed Computing Application Programming Interface
- Allows easy implementation and deployment of distributed applications on multiple grid environments
- An API that hides the details of BOINC
- The DC-API is designed to be simple and easy to use
- Supports a master-worker programming model
- Work units are sequential applications

DC-API Functionality

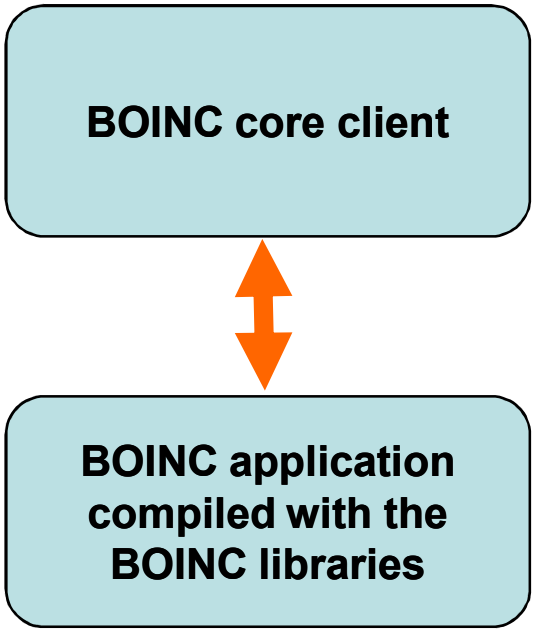


BOINC Wrapper Technology

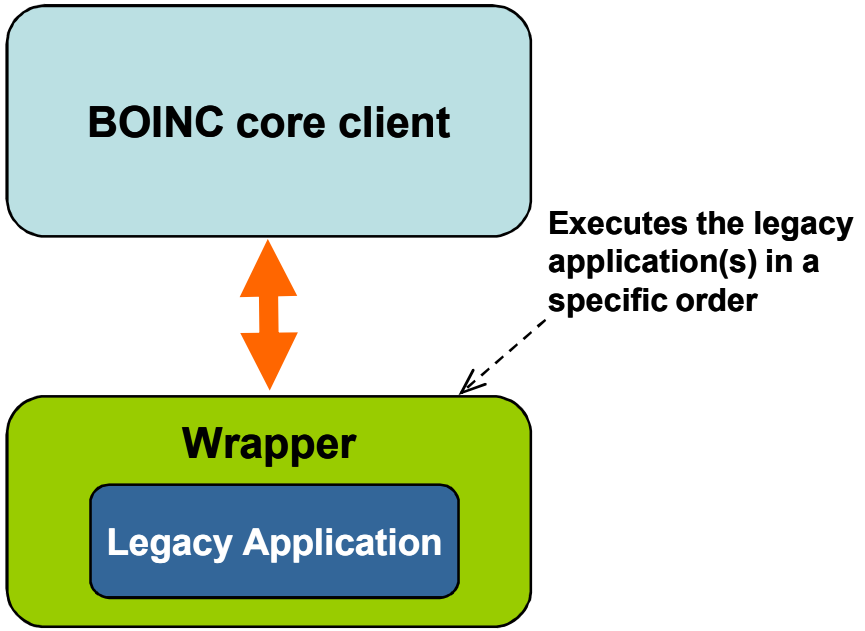
- Can be used to port legacy applications easily
- No need the change the original application
- The wrapper runs legacy applications as sub-processes
- The wrapper plays the role of the BOINC client application
- Handles all communication with the core client (fraction, CPU time reporting, suspend, resume, abort, etc.)
- Tasks:
  - Compile the Wrapper
  - Create a Job description file (XML)

BOINC Wrapper Structure

Without the Wrapper



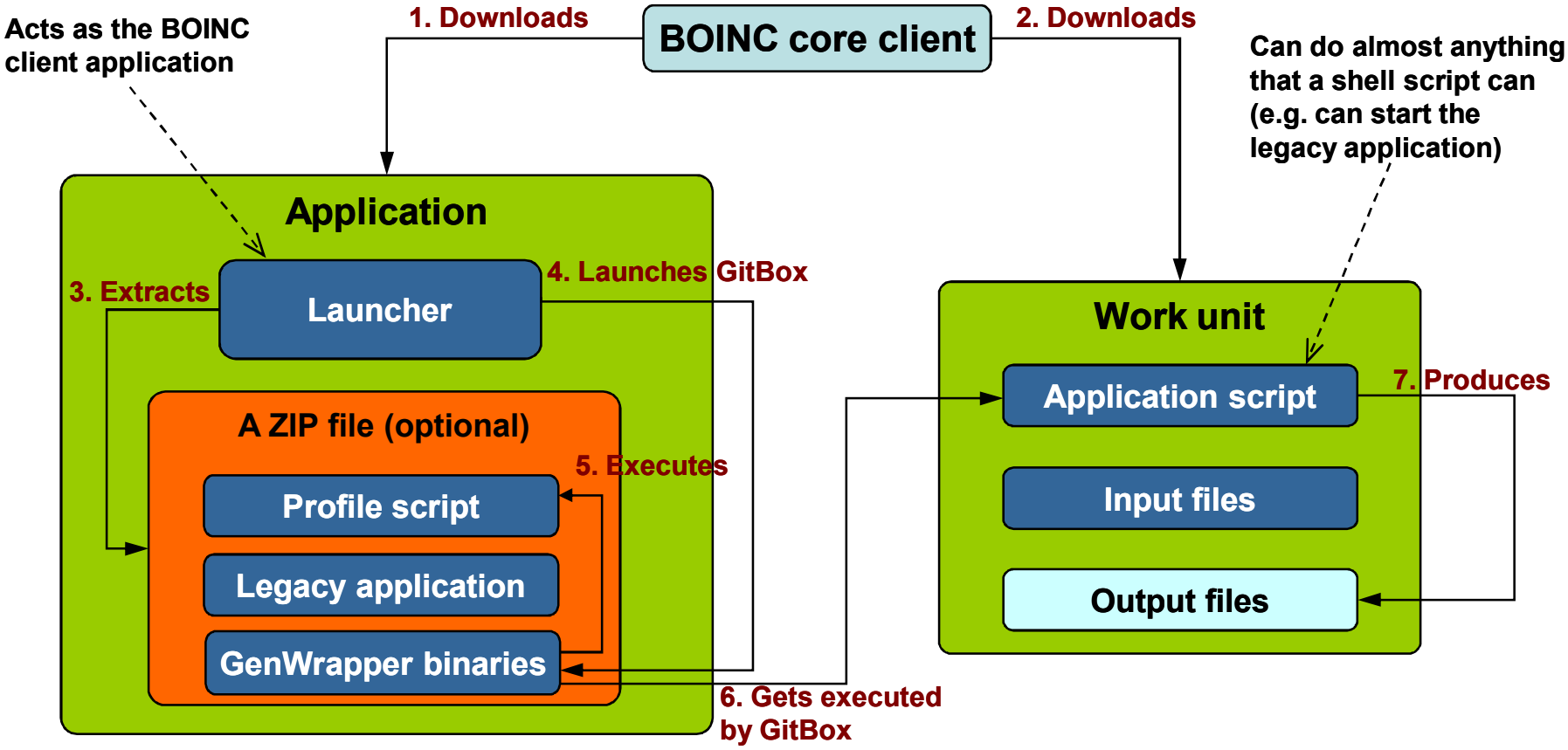
With the Wrapper



GenWrapper technology

- Generic wrapper interpreter based on *GitBox*
- Runs legacy applications without BOINCification
- Makes the BOINC API / DC-API available in POSIX shell scripting
- A shell interpreter is started instead of the real application that executes an application script
- The script
  - realizes BOINCification through script commands
  - may run legacy applications in any way
  - may perform any preparation on input-, output files, environment, etc.
  - may do whatever you can do in a script

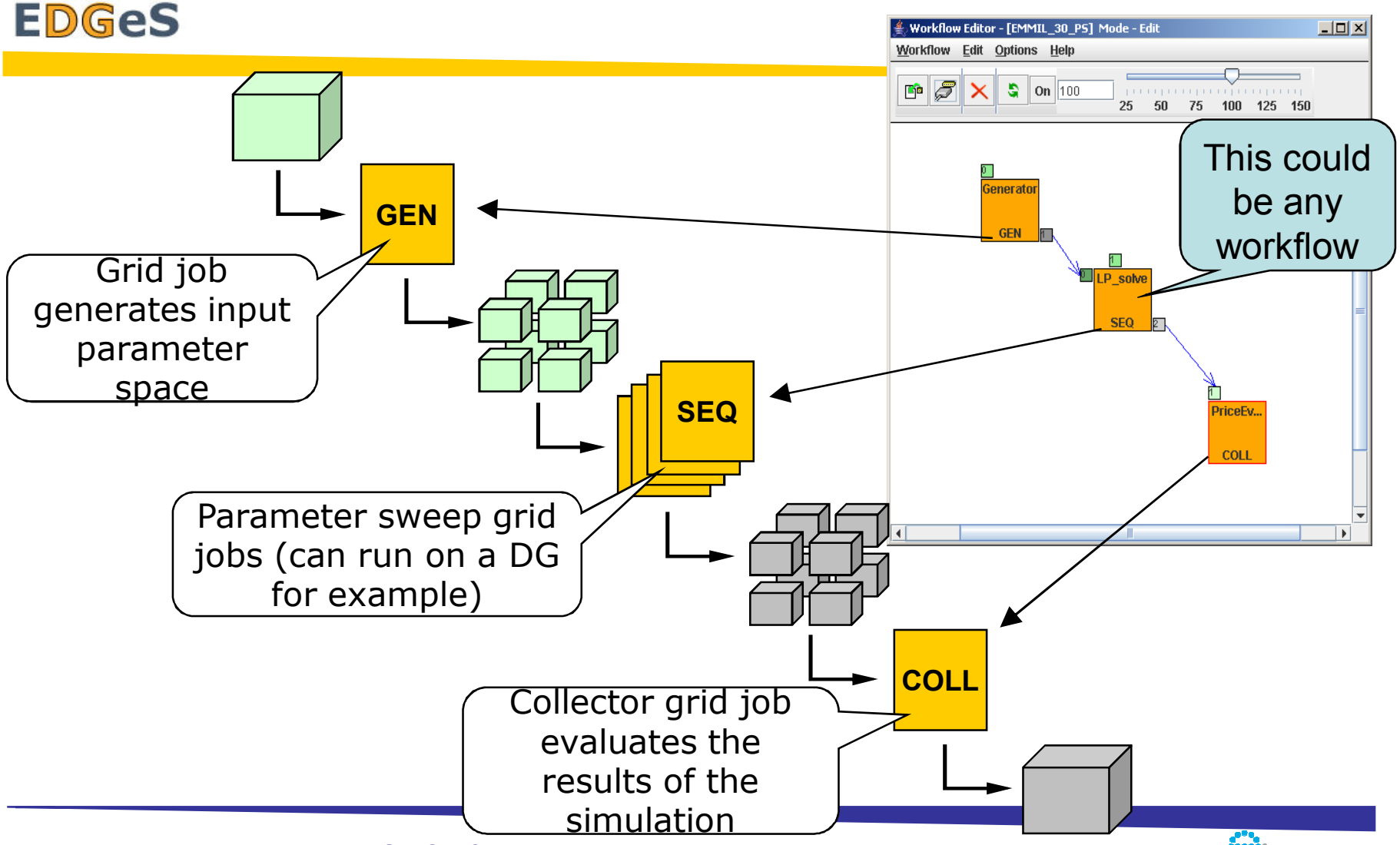
GenWrapper Structure



gUSE & WS-PGRADE

- gUSE (grid User Support Environment)
  - is a grid virtualization environment
  - exposes the grid as a workflow
  - enables the execution of workflows simultaneously in many grids no matter what their middleware is
- WS-PGRADE is the user interface to support
  - editing, configuring, publishing workflows (as grid applications)
- Workflows can be built from four types of jobs
  - Normal, sequential job
  - Generator
  - PS (Parameter Sweep)
  - Collector
- Jobs can be submitted to a BOINC DG

gUSE Parameter Study Workflow



Comparison of the methods, APIs, and technologies (server side)

	BOINC API	DC-API	gUSE
Programming language	C/C++/Fortran/Python	C/C++(/Python)	Jobs can execute any kind of binaries
Number of applications to be written	3 (WU generator, Validator, Assimilator)	2 (Master application, Validator)	1 Workflow with at least 2 jobs (Generator, Collector)
Easy to use GUI	✘	✘	✓
Can be used with other Grid middlewares	✘	✓	✓
Implementation effort (1. Maximum, 3. Minimum)	1	2	2.5
Programming concept	Master-worker	Master-worker	Workflow based
Parameter study support	✘	✘	✓
Method of execution	From the command line	From the command line	From a portal

Comparison of the methods, APIs, and technologies (client side)

	BOINC API	DC-API	BOINC Wrapper	GenWrapper
Programming language	C/C++ /Fortran /Python	C/C++ (/Java, experimental /Python)	No programming, only a job description file	POSIX shell scripting
Suitable for legacy applications	✓ (difficult implementation)	✓ (difficult implementation)	✓	✓
Suitable for non-legacy applications	✓	✓	✗	✓ (if it is based on shell scripts)
Application specific configuration support	✗	✓	✗	✓
Logging support	✗	✓	✗	✓
Checkpointing support (application level)	✓	✓	✓	✓

Resources

- BOINC API (BOINC Wiki):
  - http://boinc.berkeley.edu/trac/wiki/ProjectMain
- DC-API:
  - http://desktopgrid.hu
- gUSE:
  - http://www.guse.hu
- BOINC Wrapper:
  - http://boinc.berkeley.edu/trac/wiki/WrapperApp
- GenWrapper:
  - http://sanjuro.lpds.sztaki.hu/genwrapper