Visu@lGrid : A Model Driven Engineering Approach for BOINC

Christian Benjamin Ries

E-Mail: cbr@fh-bielefeld.de
Website: www.visualgrid.org

BOINC Workshop 2011, Hannover, 18th August 2011

http://spin.fh-bielefeld.de
Outline

• Introduction <<briefly>>

• Visu@Igrid, what happened recently, Technologies, Use-Cases

• Conclusion

• Further ideas and contributions in [close] future...
Where are we currently?

1st Programming Languages: Assembly, Fortran, C/C++

Hardwork: Bits-Bytes

Sequential, Threads, Parallel, Single-/Multi-Core, GPU, μC

(Distributed, Volunteer, Grid, Cloud) - Computing
Visu@lGrid

BOINC can be used in different scenarios:

- native application [we have do everything manually],

- native application [but multithreaded, a little bit more work has to be done for thread handling],

- on different platforms and architectures,

- single-/multi-core, GPU,

- legacy applications,

- synchronous, asynchronous messages,

- ...some more...
Visu@lGrid

By the way...

We created a wrapper for:

- **COMSOL Multiphysics**, and


- **Scilab**
Visu@lGrid

All scenarios need experience in one or more technology fields!

BOINC uses currently following technologies (could be extended):

- C/C++ programming language,
- Python programming language,
- BASH & CSH shell scripts,
- PHP for some monitoring, maintenance, and website elements,
- Perl (e.g. rBOINC),
- SQL (database queries),
- XML (configuration and RPC-requests),
- OpenCL & CUDA,
- ...some more!
Visu@lGrid

Boah... heavy...

Hard for beginners to create a BOINC project!
Visu@lGrid: First step!

We created a Model-View-Controller (MVC) framework.

Here, we have created an Object-orientated framework.

Other BOINC projects did this as well.

class Uppercase : public VisualGrid::Controller {
private:
    VisualGrid::Int *filepos;
inline void initVariants() {
    filepos = new VisualGrid::Int(0, "filepos");
}

inline void cleanVariants() {
    if(filepos != NULL) delete filepos;
}

public:
    Uppercase();
    Uppercase(const Uppercase& uppercase);
    ~Uppercase();
    void doWork();
    void doSharing();
};

try {
    VisualGrid::Boinc::Handler *handler =
        new VisualGrid::Boinc::Handler();
    handler->init();
    controller = new VisualGrid::Controller();

    // Files
    controller->addInputFile("in", "in");
    controller->addOutputFile("out", "out");
    controller->addHandler(handler);
    controller->start();
} catch(VisualGrid::Exceptions::Exception ex) {
    ...
}
Visu@lGrid : Second step!

We created a Model-View-Controller (MVC) framework and coupled it with an Aspect-oriented framework.

Idea & Question: Dynamic creation of scientific applications during project runtime. Any pros in relation to different configured workunits? Is it a good approach to use it within Visu@lGrid?

In progress...
Visu@lGrid: Third step!

We created an UML Profile:

Christian Benjamin Ries et al., UML Profile for Berkeley Open Infrastructure for Network Computing (BOINC), to be published
Visu@lGrid: Third step!

This UML Profile could be applied to UML diagrams:

C.B.Ries, C.Schröder, and V.Grout, A Tree Model and an Integrated Development Environment Concept for Visu@lGrid, to be published, Seventh Collaborative Research Symposium on Security, E-learning, Internet and Networking (SEIN 2011)

BOINC Workshop 2011 (H) · 17.08.11 · Christian Benjamin Ries, University of Applied Sciences Bielefeld
Visu@lGrid: Forth step!

We created some Domain-specific languages, e.g.

```
base Base {
    attr name : QString;
}

Project -> Base {
    //attr projectName : QString; // Provided by Base::name.
    attr id : QString;
    attr workingDirectory : QString;
    attr state : bool;
    association operations : UserOperation [*];
    association permission : PermissionAssignment [1];
    association applications : Application [*];
    association hosts : Host [*];
    association sans : StorageAreaNetwork [*];
}

Host -> Base {
    //attr hostname : QString; // Provided by Base::name.
    attr operatingSystem : enum { Linux32, Linux64 };
    attr main : bool;
    //association firewall : Class [1];
    association permission : PermissionAssignment [1];
    association services : Service [*];
    association nics : NetworkInterfaceCards [*];
    association nicbound : NetworkInterfaceBound [*];
    association exports : PortExport [*];
    association imports : PortImport [*];
}
```

Distributions {
    //serverLinux32: "~/ubuntu-10.1-i386.iso";
    serverLinux64: "~/ubuntu-10.1amd64.iso'';
}

Project {
    maintainer: "Christian Benjamin Ries";
    contact: "cbr@fh-bielefeld.de";
    name: "visualgrid";
}

Deployment

Science App.

```cpp
int a = 42;
if(a > 42)
    modeledFunction(a);
else a = 42;
```

```cpp
worker Spinhenge {
    wrapper("Matlab", "Argv[1],
             Argv[2]", weight, checkpointFile,
             fractionDoneFile, ...));
}
```
Visu@lGrid : Pros

Pros:

- Code-Generation will produce always valid and executable code!

- Platform-independent (PIM) and Platform-specific (PSM) models are definable!
Visu@IGrid : Cons

Cons:

- Current tools are heavy and difficult to use, e.g. Eclipse.
  -> Here, I can program and love programming in C/C++, as a result I have some problems with Java and Eclipse, BUT, ~80% of work to have a MDE environment and tool facilities is done and spend for Eclipse!

- Some work have to be spend for new tools in C/C++. 
Visu@lGrid : We have the time :)
The Challenge...

... make it as easy as possible!

The main goal of Visu@lGrid is the realization of an integrated development environment which allows one to develop applications based on the “Berkeley Open Infrastructure for Network Computing (BOINC)” by graphical and textual modeling and complete code generation within a Model Driven Engineering process!
Conclusion

- It will work!

- Visu@lGrid will be an **easy to use** way to **create**, **deploy** and **maintain** a BOINC based project!

- Visu@lGrid concepts can be used in additional scope of applications:
  - Message Passing Interface (MPI)
  - [hybrids] MPI and OpenMP
  - Set-up of Grid- and Cloud-Computing environments

- Open protocols allow to exchange configurations and whole BOINC projects.
Spring 2012

Your contributions are welcome!
Thank you,


Telefon: +49 (0) 521 106 71222
e-Mail: cbr@fh-bielefeld.de
Website: www.visualgrid.org

Computational Materials Science & Engineering (CMSE)
Room 202, Werner-Bock-Straße 36
33602 D-Bielefeld