Approach of a UML Profile for the Berkeley Open Infrastructure for Network Computing (BOINC)

Christian Benjamin Ries
Christian Schröder, and Vic Grout

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

IEEE Conference on Computer Applications and Industrial Electronics (ICCAIE)
Penang (Malaysia), 4\textsuperscript{th} - 7\textsuperscript{th} December 2011
Outline

- BOINC – what's that?
- UML Profile for BOINC
- Conclusion & Future Work…
BOINC is Public Resource Computing
BOINC - Public Resource Computing

- [1] **Scientific Application** (Independent Software Vendor, Legacy-applications, personal creative implementations...)
- [1..*] sets of runtime-parameter
- [*] additional files
BOINC's Architecture - Overview

- minimum of 6 components on server-side required
- they track the lifetime of work and handle results
- others are optionally, and more could be added
BOINC work life-time

- Work generator: Create work-units
- Transitioner: Create WUs
- Feeder: Send WU to shared memory
- Worker: Request for computation
- CGI Scheduler: Schedule WU
- Transitioner: Set WU in progress
- Worker: Return WU as a result

Client's software

- Quorum: Valid
- Validator: Validate results
- Prepare result for validation
- Error: no
- Transitioner: Upload files
- File uploader: Workers joining and leaving

Could be modified, own implementations are possible.
BOINC's Architecture - Overview

- any BOINC component could be started on a different host
- main host “vg-challenge-01” needs access to any host
- all hosts needs access to projects database
BOINC

All scenarios need experience in more than one technology fields!

Currently BOINC uses following technologies *(is expansible!)*:

- C/C++ programming language
- Python programming language
- BASH & CSH shell scripts
- PHP *[monitoring, maintenance, web, …]*
- Perl *[e.g. rBOINC]*
- SQL *[database queries]*
- XML *[configuration and RPC-requests]*
- GPU calculations with OpenCL & CUDA
- …some more!

...can be used in different scenarios:

- native application *[your impl.]*
- native application *[multithreaded, a little bit more work has to be done for thread handling]*
- different platforms, architectures
- single-/ multi-core, GPU
- legacy-applications
- synchronous, asynchronous msg.
- different working groups *[scientists, administrators, …]*
UML Profile for BOINC

- UML profile is a kind of UML extension mechanism
- It specializes language elements, describes constraints
- Based on three basic mechanisms:
  1. stereotypes
  2. tagged values
  3. constraints

Case-Study: LMBoinc

LMBoinc - BOINC 'Hello World!' Example: This project is a test BOINC project to manipulate video streams with different image manipulation algorithms.

Case-Study: LMBoinc

- Infrastructure definition

ICCAIE2011, 4.-7. December 2011, Penang (Malaysia)
Case-Study: LMBoinc

- Infrastructure definition
- Connection between hosts, file-sharing definition
Case-Study: LMBoinc

- Infrastructure definition
- Connection between hosts, file-sharing definition
- Role-Based Access Control (RBAC) definition, can be specific your each element
- Infrastructure definition
- Connection between hosts, file-sharing definition
- Role-Based Access Control (RBAC) definition, can be specific your each element
- Scientific application and server component logic
Case-Study: LMBoinc

Specify input video data

Compute with different algorithm!


ICCAIE2011, 4.-7. December 2011, Penang (Malaysia)
Future Work

➔ How can we model input data (so-called workunits)?

➔ How can we track the life-time of one workunit, and how can we handle exceptions on specific workunits?

➔ How can we model a scientific application with whole support of BOINC functionalities, e.g. atomic functions and checkpointing?

➔ BOINC does support asynchronous messages, how can we model these for individual communications, e.g. long-running computations should not abort after a while.

➔ Scripts for autonomous maintaining of one BOINC projects can be added and should be called periodically. How can a timing diagram look like?
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