BoincLite: A thin BOINC client library

Peter Hanappe
Sony Computer Science Laboratory
hanappe@csl.sony.fr

December 17, 2008

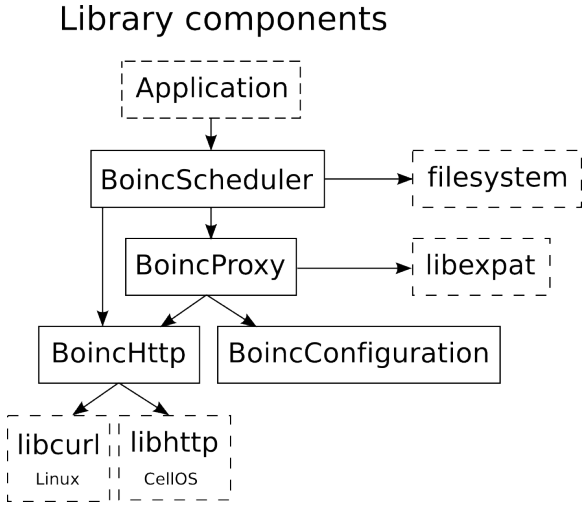
BoincLite defines the minimum API to realise a simple BOINC client. Its main functionality is to download work units, to upload the results, and to make sure there is always one work unit ready to be executed. It has a fair number of restrictions compared to the existing BOINC manager:

- It handles only one BOINC project.
- It doesn't manage the resource usage (CPU, memory, disk) nor the scheduling between projects and work units. If resource management is needed, the calling application will have to implement it.
- It doesn't provide any glue for constructing a user interface. Again, this has to be managed by the calling application.
- Only one workunit is computed at once and at most one workunit is waiting to be executed.
- The downloads and uploads are performed sequentialy.
- The library doesn't use multi-threading. However, it is straighforward to spin of separate threads for the scheduler for and the computation of the work unit so that they run parallel to the main thread.

The library consists of four components: BoincScheduler, BoincProxy, BoincConfiguration, and BoincHttp (see Fig.).

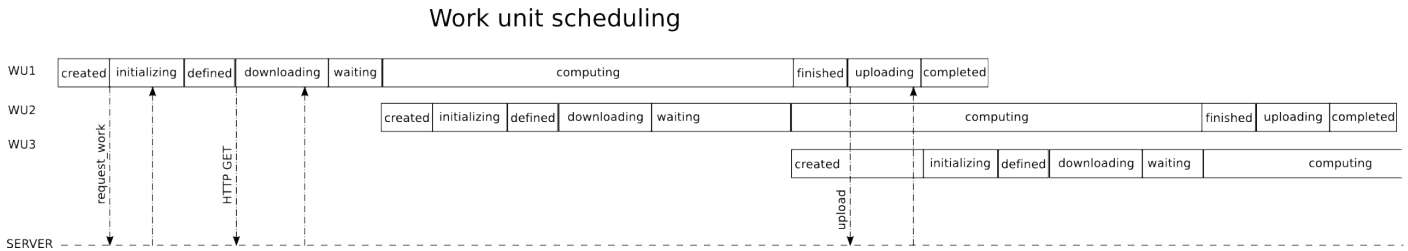
BoincHttp is a thin abstraction layer for handling HTTP requests. It is used as a portable layer that can be easily implemented on top of existing HTTP libraries, such as libcurl on Linux or libhttp on CellOS.

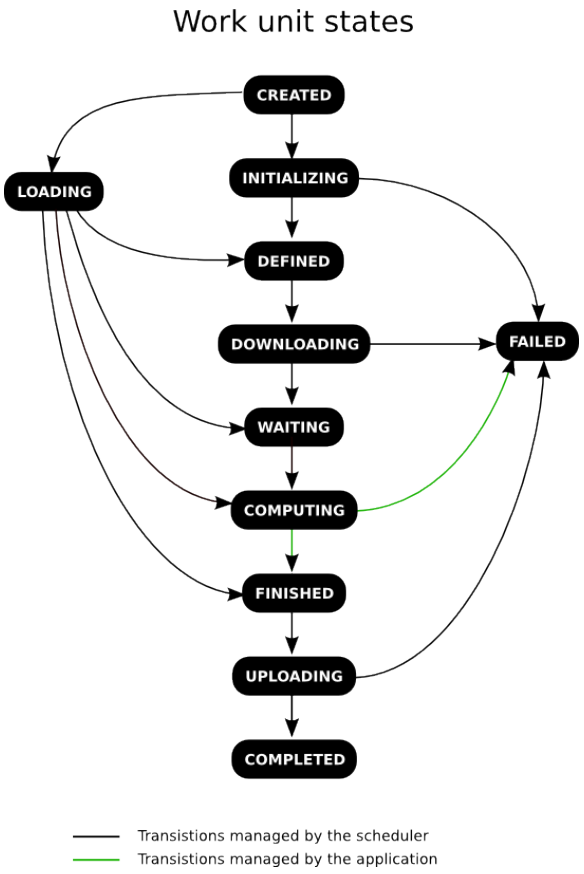
The BoincConfiguration is another shallow abstraction layer. It manages all the configuration data needed by BOINC, such as the user ID, the authenticator string, but also the disk usage or the FLOPS of the host machine.



BoincProxy handles the RPC messages with the BOINC server. It uses expat to parse the XML data. It relies on BoincHttp and BoincConfiguration for all the platform specific functions.

The principal functionality of the BoincScheduler is assure that there is always one work unit available to be computed. It is possible to overlap the computation with the network transfers but it requires threading support from the calling application. The scheduler also handles the loading/storing of work units from/to disk. The BoincScheduler calls upon the BoincProxy to request new work units or to upload the results and uses BoincHttp to download the data of the work unit.


Work unit scheduling
WU1
WU2
WU3
SERVER
created
initializing
defined
downloading
waiting
computing
finished
uploading
completed
request work
HTTP GET
upload


Work unit states
CREATED
LOADING
INITIALIZING
DEFINED
DOWNLOADING
FAILED
WAITING
COMPUTING
FINISHED
UPLOADING
COMPLETED
Transistions managed by the scheduler
Transistions managed by the application