Developing BOINC applications

Daniel Lombraña González

March 5, 2010

Outline

Applications

BOINC

- is a framework.
- is developed in C++.
- is open source (LGPLv3).
- provides an API.

Application Programming Interface

BOINC's API

is a set of C++ functions.

Note

Most of the functions and methods are written in C, so it is possible to employ them from other programming languages.

Functions

- Functions return an error code of type *integer*.
- Zero, means success.

Start and End

Firstly, we have to initialize the BOINC application:

Initialization Function

```
int boinc_init();
```

When the application has finished, we have to call the finish function.

Finish Function

```
int boinc_finish(int status);
```

File Names

Applications that use I/O files, have to employ the following function:

File name resolve function

int boinc_resolve_filename(char *logical_name, char *physical_name, int len);

File name resolve function

int boinc_resolve_filename_s(char *logical_name, std::string& physical_name);

Example

Instead of using this:

Standard Function

f = fopen(“my_file”,“r”);

We will use:

Opening a file in BOINC

```
string resolved_name;
retval = boinc_resolve_filename_s("my_file", resolved_name);
if (retval) fail("can't resolve filename");
f = fopen(resolved_name.c_str(), "r");
```

I/O wrappers

Porting applications to BOINC require to change all I/O file functions *fopen()* by BOINC ones:

BOINC function

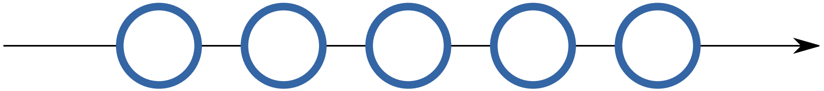
boinc_fopen(char* path, char* mode);

This function is independent from OS platforms (Microsoft Windows, GNU/Linux and MacOSX).

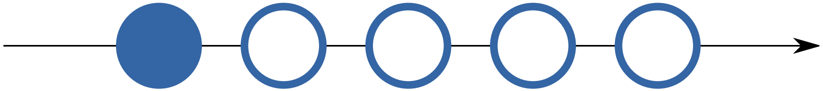
Checkpointing

- Applications with long times to solution usually will want to save intermediate execution points.
- These points should have all the necessary information to restore the computation from the last saved point.

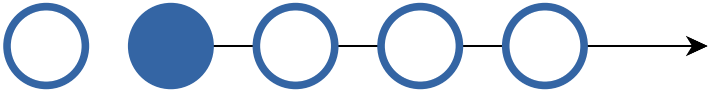
Checkpointing



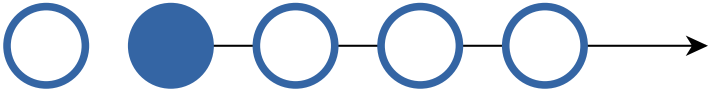
Checkpointing



Checkpointing



Checkpointing



Functions

Starting Checkpointing

```
int boinc_time_to_checkpoint();
```

This function can be used as many times as needed.

Finishing Checkpointing

```
void boinc_checkpoint_completed();
```

Critical code

- There are parts of an application that are critical in its execution.
- For this reason, we do not want to stop the execution of those parts at any moment.
- BOINC provides several functions to assure the execution of any critical section without interruptions.

Functions

Starting the critical section

void boinc_begin_critical_section();

Ending the critical section

void boinc_end_critical_section();

Note

This is carried out automatically in the checkpoints.

Progress Bar

The BOINC's client shows the percentage of carried out work. To update that percentage bar, we have to use the following:

Function

```
boinc_fraction_done(double fraction_done);
```

The next function obtains the last computed percentage:

Function

```
double boinc_get_fraction_done();
```

Gathering information from the Client

The following functions obtain information from the client:

Functions

int boinc_get_init_data_p(APP_INIT_DATA*); int
boinc_get_init_data(APP_INIT_DATA&);

Obtained data

- *core version*: Client's version in digits.
- *app_name*: Application name.
- *project_preferences*: A XML text with the preferences of the user for the project.
- *user_name*: User name of the project.
- *team_name*: Team name of the user.
- *project_dir*: Absolute path of the project folder.
- *boinc_dir*: Absolute path of BOINC root's folder.
- *wu_name*: The name of the Work Unit.
- *authenticator*: The authenticator for this project.
- *slot*: Slot number.
- *user_total_credit*: Total credif of this user for the project.
- *user_expavg_credit*: Average credit of the user per day.
- *team_total_credit*: Total credit of the team for this project.
- *team_expavg_credit*: Average credit of the team per day.
- *host_info*: A struct describing the HW and OS of the host.

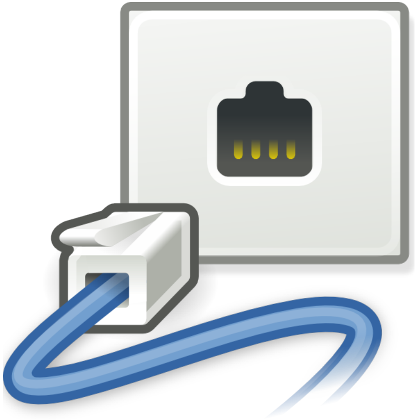
Checking the application

- BOINC allows to run applications without the client.
- The next function enables the “standalone”:

Standalone Function

```
int boinc_is_standalone(void);
```

Networking



Because of User Preferences

Warning the user

The next function warns the user asking for allowance to connect to the network:

Función

void boinc_need_network();

Checking the network

The next function checks if the application can go on-line:

Función

int boinc_network_poll();

Finishing the communications

When communications have been finished, we have to call the following function:

Function

void boinc_network_done();

Acknowledgments

Icons from the Tango Desktop project and Gnome Desktop (Creative Commons & GPL License)