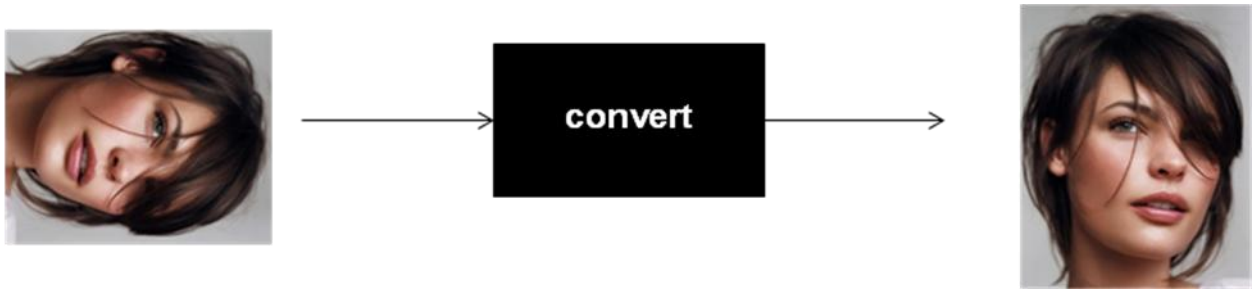
Desktop Grid Training – 2012 Mainz, Germany

Porting applications with GenWrapper, submitting tasks with 3G Bridge

In this tutorial you will be porting the `convert` application to the BOINC platform using the GenWrapper technology. The convert application is a simple command line tool that enables the user to create image transformations without the need of using graphical user interface. We will see how to run the application in a Desktop Grid platform without the need of understanding or having the source code.

We will deploy 3G Bridge and use its WebService interface to (remotely) submit tasks for our application.



This tutorial assumes that you created a BOINC project named *'test'* in the previous tutorial and successfully connected your BOINC Client to it.

Perform the following steps carefully in the Virtual Machine and ask for support from the tutors when required:

- **Install ImageMagick**
  - `apt-get install imagemagick`
  - The application (`convert`) we will port to the Desktop Grid relies on ImageMagick. Normally we would include ImageMagick in the DG application bundle and make it deploy locally for each work unit, but since this is a training we are simplifying things and installing it ☺
- **Change to the BOINC project user you created earlier:**
  - `su - boinc-test`
- **Download the latest version of GenWrapper:**

- 32bit Linux:
  - *wget --no-check-certificate https://sourceforge.net/projects/genwrapper/files/genwrapper-linux32-2508.tar.gz*
- (64bit Linux:
  - *wget –no-check-certificate https://sourceforge.net/projects/genwrapper/files/genwrapper-linux64-2508.tar.gz*)

- **Unpack the archive:**
  - *tar –xzvf genwrapper-linux32-2508.tar.gz*
  - Rename the `gw_launcher` to `testapp`
  - Use *'i686-pc-linux-gnu'* as platform name
- **Download the convert application bundle from the following location:**
  - *wget http://kanuka.lpds.sztaki.hu/training/convert32.zip*
  - rename it to `convert.zip`
- **Create the application script (`wu_script.sh`), open your preferred (available) text editor and enter the following – please do not copy&paste since invisible line-ending characters can cause nasty things - :**

```
set -xv

INPUT_LOGICAL_FILE_NAME="input.jpg"
OUTPUT_LOGICAL_FILE_NAME="output.jpg"
ZIPFILE_LOGICAL_FILE_NAME="convert.zip"

INPUT_FILE_NAME=`boinc resolve_filename "${INPUT_LOGICAL_FILE_NAME}"`
OUTPUT_FILE_NAME=`boinc resolve_filename "${OUTPUT_LOGICAL_FILE_NAME}"`
ZIPFILE_FILE_NAME=`boinc resolve_filename "${ZIPFILE_LOGICAL_FILE_NAME}"`

unzip ${ZIPFILE_FILE_NAME}

chmod +x ./convert
./convert -rotate 90 ${INPUT_FILE_NAME} ${OUTPUT_FILE_NAME} 1>&2


boinc finish 0
```

- **Download a jpg file for testing purposes, e.g.,**
  - *wget http://kanuka.lpds.sztaki.hu/training/input.jpg*
  - **Name it *input.jpg***
- **Put together the files into the same directory. You must have the following files:**

- `testapp` (*gw_launcher* renamed)
- `gitbox` (from the Genwrapper archive)
- `convert.zip` (contains convert)
- `client_script.sh` – our new profile script
- `input.jpg` – input file

- **Test the application in standalone mode.** (Standalone mode provides the possibility to test your boinc application without deploying it to a Desktop Grid environment)
  - Type: `./testapp`
  - In the case of the successful run the output.jpg is generated
  - If something is wrong you will find the error log in the dc_stderr.txt file
- **Put together your files into the following directory structure:**
  - **bundle**
    - **client**
      - **i686-pc-linux-gnu**
        - `gitbox`
        - `testapp`
        - `convert.zip`
        - `wu_script.sh`
      - testapp_client.xml (Don't worry about this you will create it later :)
    - **master**
      - testapp_master.xml (Don't worry about this you will create it later :)
- **Create the** `testapp_client.xml` **with your preferred editor – please do not copy&paste:**

```
<?xml version="1.0" standalone="no"?>
<client>
    <name>testapp</name>
    <user_friendly_name>Test Application</user_friendly_name>
    <version>1.00</version>
```

```
        <platform>
            <name>i686-pc-linux-gnu</name>
            <binary>i686-pc-linux-gnu/testapp</binary>
            <lib>i686-pc-linux-gnu/convert.zip</lib>
            <lib>i686-pc-linux-gnu/wu_script.sh</lib>
            <lib>i686-pc-linux-gnu/gitbox</lib>
        </platform>
    </client>
```

- **Create the** `testapp_master.xml` **with your preferred editor:**

```
<?xml version="1.0" standalone="no"?>
<master>
    <name>testapp</name>
    <version>1.00</version>
    <daemon>
       <name>sample_trivial_validator</name>
       <arguments>
         <arg>-d</arg>
         <arg>3</arg>
         <arg>-app</arg>
         <arg>testapp</arg>
       </arguments>
  </daemon>
</master>
```

- **Deploy the bundle under the boinc_test user:**
  - Add the following to `~/project/project.xml` just before the closing `</boinc>` tag:

```
<app>
   <name>testapp</name>
   <user_friendly_name> Test Application </user_friendly_name>
</app>
```

  - run the `xadd` command
  - Deploy the daemons:
    `boinc_appmgr --add --master master/testapp_master.xml`
  - Deploy the client type:
    `boinc_appmgr --add --client client/testapp_client.xml`
  - Stop the server: `stop`
  - Restart the server: `start`

- In case you have trouble the bundle can be downloaded from *http://kanuka.lpds.sztaki.hu/training/testapp-i686-pc-linux-gnu-bundle.tar.gz*

- **Create new directory for the master application**:
  - `cd ~`
  - `cd master`
  - `mkdir testapp`

- `cd testapp`
- Create working directory: `mkdir workdir`
- Copy you input.jpg file to the workdir

- **Download the source of the master application (to the testapp directory)**:
  - *wget http://kanuka.lpds.sztaki.hu/training/master.tar.gz*
  - Extract it: `tar –zvxf master.tar.gz`
- **Switch to root and install necessary libraries for compilation:**
  - `exit`
  - `apt-get install libdcapi-boinc-dev`
  - `apt-get install make`
  - `apt-get install g++`
  - Swtich back to the boinc-project user e.g: `su - boinc-test`
- **Compile the master application:**
  - `make`
- **Edit the** `master.conf` **file**:
  - `master.conf` contains some configuration options so that the master application knows where the BOINC project is as well as some application settings (e.g., maximum size of outputs, whether to use redundant computing, allowed maximum used memory, etc.).
  - Make sure that all paths (WorkingDirectory, WorkingDirectory, BoincConfigXML) point to inside your project: `/var/lib/boinc/`<YOU PROJECT SHORT NAME>`/...` – e.g., `/var/lib/boinc/test/...` If your projects short name is '**test**' and your application name is '**testapp**', the directory names should look like this:
    - `WorkingDirectory=` `/var/lib/boinc/`**`test`**`/master/`**`testapp`**`/workdir`
    - `BoincConfigXML =` `/var/lib/boinc/`**`test`**`/project/config.xml`
    - `ProjectRootDir = /var/lib/boinc/`**`test`**`/project`
- **Start the master application:**
  - `./master –c master.conf`

  - If you configured everything ok in the previous step it will create a work unit and the application will **not terminate**, but wait for results.

- **Now start you BOINC client application**
  - `boincmgr`
  - Wait until the client picks up the work and completes it. To speed up the process you can select your project (on the Projects tab) and click update, so the client will ask for work immediately.
  - When the client finished the work the output file (output.jpg) will be transferred to the server and copied to the work directory and the master application will **terminate**.

- **Optional exercise:**
  - If you successfully completed all the steps you can try to modify the client application to do different kind of image transformation. After modifying the application script you will need to increase the application version testapp_client.xml and redeploy the application. When you have the new client version installed start the master application and see what happens with your image.

Job submission via 3G Bridge

Instead of an application specific master (used previously), we will use a generic interface: 3G Bridge and its WebService (SOAP) interface to submit jobs to our application.

First we need to deploy and configure it:

- Install **3G Bridge** and **wsclient**
  - `apt-get install 3g-bridge 3g-bridge-wsclient`
  - When asked whether *"Enable running the 3G Bridge in stand-alone mode?"* – select **No**.

- **Switch to the BOINC user**, by e.g., `su - boinc-test`

- **Import the 3G Bridge database schema** (replace `boinc_test` with your project name when needed. Note: for the db, "_" is used and not "-" in the database name):
  - `mysql boinc_test < /usr/share/3g-bridge/schema.sql`

- **Deploy 3G Bridge and wsclient** in the project using the boinc_appmgr command>
  - `boinc_appmgr -add -master /usr/share/3g-bridge/master-ws.xml`

- `cd ~/master/3g-bridge/`
- **Edit `3g-bridge.conf`** and change the following:
  - `log-level = DEBUG`
  - `monlog-target = /tmp/monitor.log`
  - In the `[database]` section set the following (replace `boinc_test` as needed):
    - `host = localhost`
    - `name = boinc_test`
    - `user = boinc_test`
    - `password = **<the value of db_passwd in ~/project/config.xml>**`
  - In the `[bridge]` section:
    - `log-target = stdout`
    - `pid-file = /var/lib/boinc/test/master/3g-bridge/3g-bridge.pid`
  - In the `[wssubmitter]` section:
    - `log-target = stdout`
    - `pid-file = /var/lib/boinc/test/master/3g-bridge/wssubmitter.pid`
    - `input-dir = /var/lib/boinc/test/master/3g-bridge/input`
    - `ouput-dir = /var/lib/boinc/test/master/3g-bridge/output`
  - We will now enable the DC-API plug-in. Scroll to the end of the file and add the following lines:

```
[SZDG]
disable = false
handler = DC-API-Single
dc-api-config = /var/lib/boinc/test/master/3g-bridge/dc-api.conf
```

  - Save and exit.
- **Create** the `/var/lib/boinc/test/master/3g-bridge/input` and `/var/lib/boinc/test/master/3g-bridge/output` directories.
- **Edit** ~/project/test.httpd.conf, and add the following to the beginning of the file:

```
Alias /3g-bridge/ /var/lib/boinc/test/master/3g-bridge/output/
```

- **Now edit** `~/master/3g-bridge/dc-api.conf` and copy everything to the end of the file from the [Client-testapp] section in ~/master/testapp/master.conf (including the [Client-testapp] line). The end of the file should look like this:

```
[Client-testapp]
name = testapp
LogLevel = Debug
Redundancy = 1
MaxOutputSize = 10 MiB
EstimatedFPOps = 11433984610301
EnableResume = false
DelayBound = 7200
```

- Finally we need to **create a queue inside 3G Bridge**, so it will know where it should send the incoming tasks (which plug-in to use):
  - `mysql boinc_test`
  - `insert into cg_algqueue (grid, alg, batchsize, statistics) values ('SZDG', 'testapp', 1, NULL);`
  - `exit;`

- **3G Bridge is almost ready to be used with our testapp application. To finish:**
  - restart the BOINC project by issuing `stop; start`
  - switch back to root and reload the apache webserver:
    - `/etc/init.d/apache2 reload`

Comments:

- BOINC puts all logs to *~/project/log_<hostname>/* directory. Each component has its own log file (e.g., 3G Bridge, wssubmitter). If something fails check there.

Job submission and management:

- We will use the wsclient command, which is a command line interface to the WebService interface of 3G Bridge ("wssubmitter").
  - wsclient can be run on any (remote) host, for this tutorial we use a single machine.
- **Create a directory and change to it**: `mkdir ~/submit; cd ~/submit`
- **Copy** *input.jpg* **to this directory**
- **Execute the following command:**

```
wsclient –e http://localhost:8091/ –m add –g SZDG –n testapp –i input.jpg=/var/lib/boinc/test/submit/input.jpg –o output.jpg
```

Itt will print out an uuid like this:

cc6eb8bf-5366-428d-9ade-6d05ab33e12d

- **You can use the uuid to query the status of the job:**

  `wsclient –e http://localhost:8091/ -m status –j cc6eb8bf-5366-428d-9ade-6d05ab33e12d`

  It will print out something like this:

  cc6eb8bf-5366-428d-9ade-6d05ab33e12d Running

- Once the application is finished (status *Finished*) you can **query the location of the output file** (and download it with `wget`):

  `wsclient –e http://localhost:8091/ -m output –j cc6eb8bf-5366-428d-9ade-6d05ab33e12d`

  It will print out something like this:

  # Output files for job "cc6eb8bf-5366-428d-9ade-6d05ab33e12d":
  output.jpg http://localhost/3g-bridge/cc/cc6eb8bf-5366-428d-9ade-6d05ab33e12d/output.jpg

- You can also **cancel a running job and or remove the output files of a finished job** from the server:

  `wsclient –e http://localhost:8091/ -m delete –j cc6eb8bf-5366-428d-9ade-6d05ab33e12d`

That's it! ☺

Extra – Manage your BOINC client from your laptop via GUI:

1. Install the BOINC client on your laptop from **http://boinc.berkeley.edu**. - *We will run applications only in the Virtual Machine, but we will use the GUI for remote managing our client in the VM*.
2. Make sure your VM uses bridged networking or host only networking (so it is accessible from you laptop).
   a. When changing network types make sure you execute "`dhclient eth0`" as root in the VM, to update the IP address of the VM.
3. Make sure the BOINC client in the VM allows remote access/ management. In the Virtual Machine:
   a. Edit **/etc/default/boinc-client** file.
   b. Uncomment the line '`#BOINC_OPTS="--allow_remote_gui_rpc`"' (remove the '#') and comment the next line (it should look like '`#BOINC_OPTS=""`').
   c. Save and exit.

d. Edit the **/etc/boinc-client/gui_rpc_auth.cfg** file, it should be empty, anyway replace its content with the word "`apple`".

e. Save and exit.

f. Execute "`/etc/init.d/boinc-client restart`" as root.

4. Connect to the BOINC client running in the VM using the BOINC Manager running in your laptop:

    a. Select from the "Advanced menu" the "Select computer..." option.

    b. Type the ip address of the virtual machine and the password set previously ("apple").

        i. You can get the IP address by executing the "ifconfig" command in the VM, it will be printed in the section "eth0" and after "inet addr:"