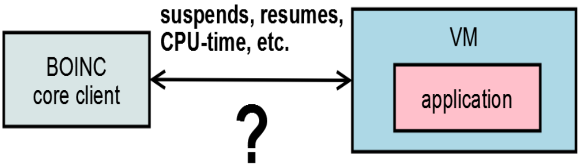
# VMwrapper

Jarno Rantala

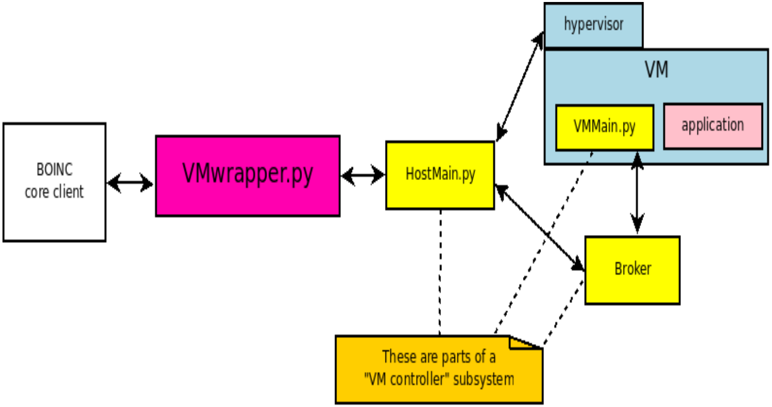
MSc student, Tampere University of Technology
CERN openlab summer student programme 09

October 23, 2009

## Introduction



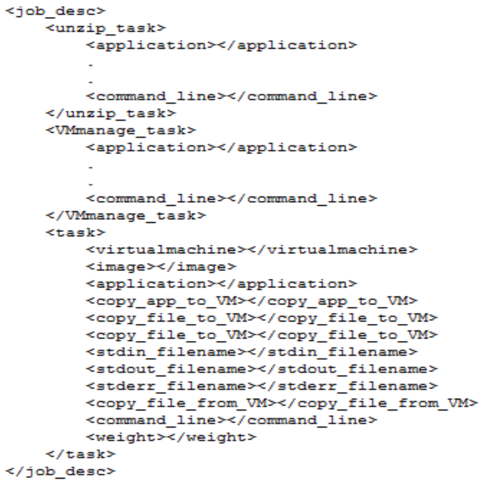
## System architecture



## VMwrapper outline

- works compatibly with the original wrapper-application
- reads job.xml file which contains a sequence of tasks which should be run
- moreover, one can now run applications in virtual machines
- written in Python using BOINC API Python bindings written by David Weir
- uses VM controllers written by David Garcia Quintas

## Format of job.xml



```
<job_desc>
    <unzip_task>
        <application></application>
        .
        .
        <command_line></command_line>
    </unzip_task>
    <VMmanage_task>
        <application></application>
        .
        .
        <command_line></command_line>
    </VMmanage_task>
    <task>
        <virtualmachine></virtualmachine>
        <image></image>
        <application></application>
        <copy_app_to_VM></copy_app_to_VM>
        <copy_file_to_VM></copy_file_to_VM>
        <copy_file_to_VM></copy_file_to_VM>
        <stdin_filename></stdin_filename>
        <stdout_filename></stdout_filename>
        <stderr_filename></stderr_filename>
        <copy_file_from_VM></copy_file_from_VM>
        <command_line></command_line>
        <weight></weight>
    </task>
</job_desc>
```

## VMwrapper details

- Defines new class called TASK
- Three different kinds of instances of the TASK: unzip task, VM manage task and task
- Unzip tasks are run before others (unpacks from project to slot dir)
- VM manage tasks are started (if there is a task using VMs)
- “Normal” tasks are run sequentially

## TASK class

- Attributes:
  application, stdin/stdout/stderr filenames, app_process (subprocess module), command line, virtual machine, ...
- Methods:
  run(), runVM(), kill(), stop(), resume(), VMrunning(), poll() ...
- Task is run on host if virtual machine -attribute is empty.

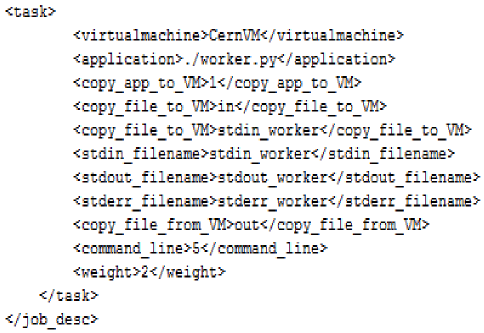
## PSEUDO code

```
[Tasks, VMmanageTasks, Unzip_tasks] = read_job_file();
run_unzip_tasks(Unzip_tasks);
start_VMmanage_tasks(VMmanageTasks) ; // if needed
for task in Tasks do
    ready = task.poll();
    while not ready do
        poll_boinc_messages();
        sleep();
        ready = task.poll();
    end
end
stop_VMmanage_tasks(VMmanageTasks);
```

## Example: worker-application

```
<job_desc>
    <unzip_task>
        <application>tar</application>
        <command_line>-xf ./cctools-2_5_2-i686-linux-2.6.tar</command_line>
        <stdout_filename>stdout_tar</stdout_filename>
        <stderr_filename>stderr_tar</stderr_filename>
    </unzip_task>
    <unzip_task>
        <application>tar</application>
        <command_line>-xf ./apache-activemq-5.2.0.tar</command_line>
        <stdout_filename>stdout_tar</stdout_filename>
        <stderr_filename>stderr_tar</stderr_filename>
    </unzip_task>
    <unzip_task>
        <application>tar</application>
        <command_line>-xf ./boincvm.tar</command_line>
    </unzip_task>
    <VMmanage_task>
        <application>./apache-activemq-5.2.0/bin/activemq</application>
        <stdin_filename></stdin_filename>
        <stdout_filename>stdout_broker</stdout_filename>
        <stderr_filename>stderr_broker</stderr_filename>
        <command_line></command_line>
    </VMmanage_task>
    <VMmanage_task>
        <application>python</application>
        <stdin_filename></stdin_filename>
        <stdout_filename>stdout_HostMain</stdout_filename>
        <stderr_filename>stderr_HostMain</stderr_filename>
        <command_line>./boincvm/HostMain.py ./boincvm/HostConfig.cfg</command_line>
    </VMmanage_task>
```

## Example: worker-application



```
<task>
        <virtualmachine>CernVM</virtualmachine>
        <application>./worker.py</application>
        <copy_app_to_VM>1</copy_app_to_VM>
        <copy_file_to_VM>in</copy_file_to_VM>
        <copy_file_to_VM>stdin_worker</copy_file_to_VM>
        <stdin_filename>stdin_worker</stdin_filename>
        <stdout_filename>stdout_worker</stdout_filename>
        <stderr_filename>stderr_worker</stderr_filename>
        <copy_file_from_VM>out</copy_file_from_VM>
        <command_line>5</command_line>
        <weight>2</weight>
    </task>
</job_desc>
```

## Remarks

- File names in command line (recognized by "./") are resolved by boinc_resolve_filename-method
- Commands in a VM are run in the same directory where VM controller is run.
- Base directory for CopyFilesToVM and CopyFilesFromVM is the home directory of user who launched VM controller in VM
- Python 2.6 is required (kill() and send_signal() of subprocess)
- VM controller must be started automatically in a VM
- VMwrapper needs BOINC API Python bindings (boinc.so)
- The "boinc" user account has to be in vboxusers-group!! (linux)

## TO DO -list

- Test other host OS's than Linux (Ubuntu 9.04)
- Decide how to compute credits in long lasting tasks (boinc_ops_cumulative, how cpu_time to ops?, needs trickle messaging)
- Implement measuring of cpu time for Windows guests (/proc/uptime is read in Linux)

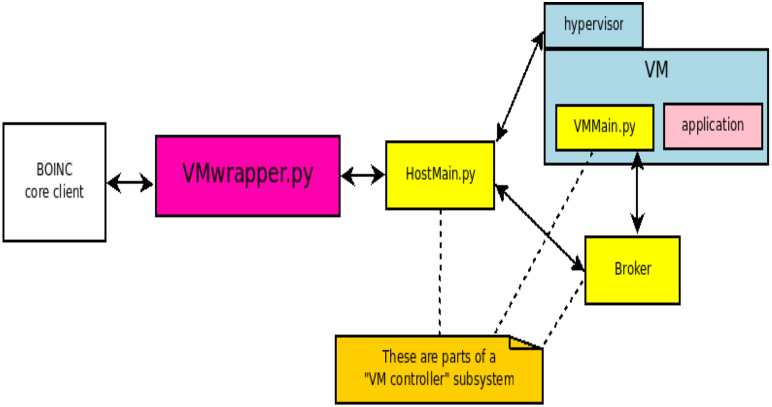
## TO DO -list

- Test if the cpu time of applications running on the host machine are measured properly.
- Test snapshotting of VM's (if we always use saveState to close VM, do we need snapshotting at all??)
- Implement stopping and resuming host applications run on Windows. (currently uses SIGSTOP and SIGCONT signals)
- Parallel tasks: one VM with one VMwrapper process, (one VM with many VMwrapper process) or should we use different VM's?

## TO DO -list

- Try to send Python runtime with the BOINC task (cx_Freeze)
- Test that we can create a VM and start it. so far the VM has already been pre-created on a host.
- Change the exit codes of VMwrapper.py

Worker-application works on Linux-host using VM!! (1 task / host)


hypervisor
VM
VMMain.py
application
BOINC core client
VMwrapper.py
HostMain.py
Broker
These are parts of a "VM controller" subsystem