TECHNICAL UNIVERSITY OF CLUJ-NAPOCA ROMANIA

GREEN CLOUD SCHEDULER

OVERVIEW

The Green Cloud Scheduler (GCS) consolidates the virtual machines in the cloud such that as few servers as possible are used and the unused servers are shut down. The controller dynamically analyzes the cloud and turns on servers as needed, turns off unused servers, deploys virtual machines and migrates virtual machines in order to free up servers. The general idea is to use as few servers as possible while maintaining the optimum load per server from the point of view of energy efficiency. An optimum load (ex. 80%) for servers can be defined and the controller will respect this value by always trying to maintain as many virtual machines on a server for fulfilling the 80% server load. For example, on a server with 10 GB memory and 1 GHz CPU, the virtual machines deployed should cumulatively request 0.8% of the CPU and memory (0.8 GHz CPU and 8 GB memory).

GREEN CLOUD SCHEDULER BEHAVIOR

GCS behavior is described in the following four steps.

1. Reads servers information by using the Java OCA API HostPool class.
2. Reads virtual machines by using the Java OCA API VirtualMachinePool class.
3. Updates the energy-aware cloud context model and takes the necessary adaptation actions.
4. Whenever a new VM is submitted to OpenNebula or an existing VM is deleted, an OpenNebula Hook event is triggered. The scheduler will react to that event and will start analyzing the cloud context for finding the optimum virtual machine placement.

The GCS is structured as a MAPE (Monitoring, Analysis, Planning and Execution) and will take the following steps for finding and enforcing new actions:

1. In the **Monitoring phase** the GCS context is updated with the new cloud information and the information returned by the OpenNebula Hooks.
2. In the **Analysis phase** the CGS will analyze the context and evaluate if new actions are necessary for improving the cloud's energy consumption.
3. In the **Planning phase** the GCS uses a learning algorithm for generating an adaptation plan consisting of deploying/migrating virtual machines and turning on/off servers actions.
4. In the **Execution phase** the GCS will use the following tools for enforcing the consolidation actions found in the planning phase:
    a. *wakeonlan* for turning ON servers
    b. *ssh* for issuing shutdown: *ssh hostname sudo /sbin/shutdown –h now*
    c. OpenNebula Java OCA API for deployment and migration of virtual machines

GAMES
Green Active Management of Energy in IT Service centres

GREEN CLOUD SCHEDULER – OPENNEBULA INTERACTION

CLOUD MONITORING INFORMATION

The GCS uses the JAVA OCA API for accessing the OpenNebula servers pool and virtual machines pool.
The Green Cloud Scheduler uses OpenNebula Java OCA API to interact with OpenNebula for

- Retrieving information about pending virtual machines form the OpenNebula VM Pool
- Retrieving information about the severs
- Enforce adaptation actions like Deploy VM, Migrate VM, Delete VM

SERVERS INFORMATION

- Because JAVA OCA API reports current CPU frequency instead of the maximum CPU frequency for a server, the maximum CPU frequency is read using cpufreq-utils. If cpufreq-utils is not installed on the OpenNebula nodes, the info from OCA is used.
- To retrieve the MAC addresses for each server defined in onehost, the GCL reads the data from ./config/arpTable and if it does not find the MAC for a server, it pings the server, then issues an arp and parses the output. The retrieved MAC address is stored in the ./config/arpTable for later use.

VIRTUAL MACHINES INFORMATION

- One issue with the VM information is that the CPU field from the VM template contains only the number of CPUs the virtual machine has requested, while the required CPU frequency is only an optional value in the REQUIREMENTS portion of the VM template. If the REQUIREMENTS portion contains information about the required CPU frequency, this information is collected and used in the deployment of the virtual machine.
- If no CPU frequency info is specified in the virtual machine template, the data from the entry VM_DEFAULT_CPU_FREQUENCY in the /config/config.properties is used.

For example, when the VM is deployed, the scheduler finds a server which can accomodate a VM requesting 3 cores on an 1950 MHz frequency CPU.

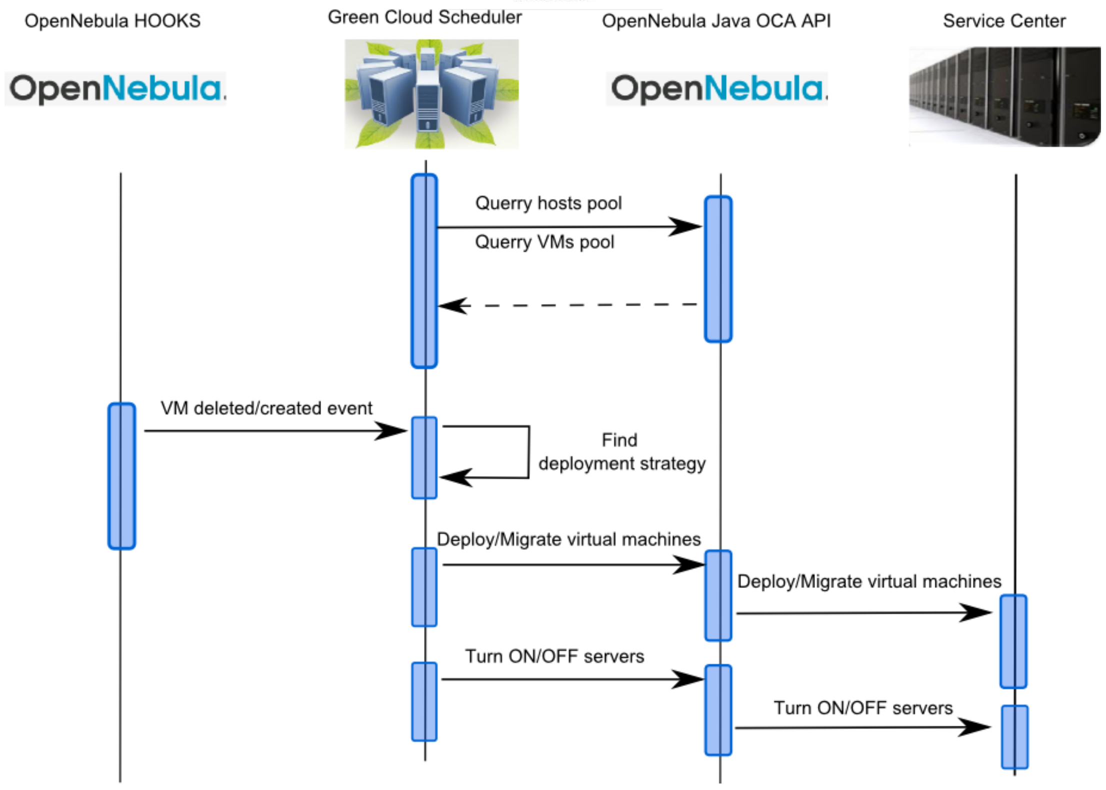
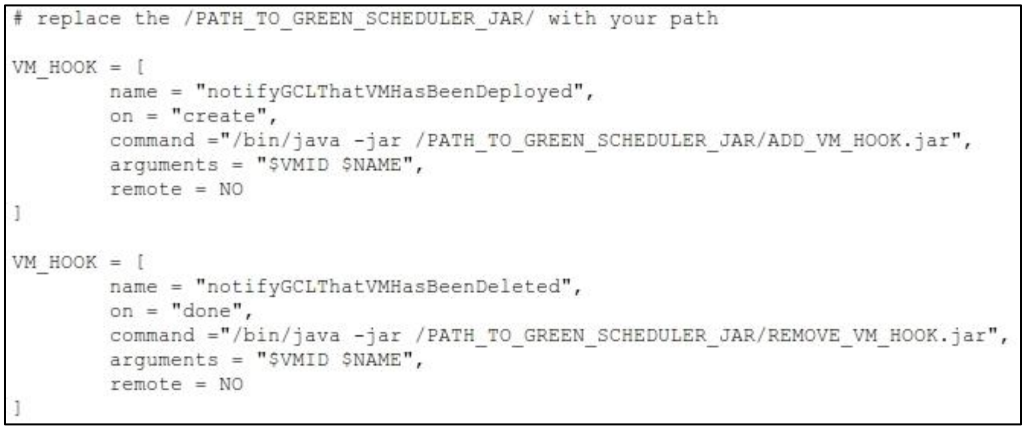


FIGURE 1:GLOBAL LOOP CONTROLLER - OPENNEBULA INTERACTION

The Green Cloud Scheduler is notified when virtual machines are created/deleted in/from the OpenNebula VM Pool by using OpenNebula hooks mechanism. The OpenNebula hooks are defined in the /etc/one/oned.conf file by entering the lines presented in the following figures.



```
# replace the /PATH_TO_GREEN_SCHEDULER_JAR/ with your path

VM_HOOK = [
        name = "notifyGCLThatVMHasBeenDeployed",
        on = "create",
        command ="/bin/java -jar /PATH_TO_GREEN_SCHEDULER_JAR/ADD_VM_HOOK.jar",
        arguments = "$VMID $NAME",
        remote = NO
]

VM_HOOK = [
        name = "notifyGCLThatVMHasBeenDeleted",
        on = "done",
        command ="/bin/java -jar /PATH_TO_GREEN_SCHEDULER_JAR/REMOVE_VM_HOOK.jar",
        arguments = "$VMID $NAME",
        remote = NO
]
```

FIGURE 2: DEFINING OPENNEBULA HOOKS IN THE /ETC/ONE/ONED.CONF FILE

GREEN CLOUD SCHEDULER INTERACTION WITH OTHER TOOLS

- ***Wakeonlan interaction***
  The Green Cloud Scheduler uses the wakeonlan tool to remotely wake up the data center servers. The following command is used: wakeonlan_tool serverMAC_Address .
- ***Cpufreq-utils interaction***
  The Green Cloud Scheduler uses the cpufreq-utils package to retrieve the maximum frequency of the target server .The following command is used: cpufreq-info -l .
- ***OpenSSH and Shutdown interaction***
  The Green Cloud Scheduler uses the OpenSSH to turn off the data center servers remotely. The following command is used for turning off servers: ssh hostname sudo /sbin/shutdown –h now To enable the remote execution of /sbin/shutdown, on each data center server the following configuration steps are required:
  - Execute : sudo chmod u+s /sbin/shutdown
  - Add “oneadmin ALL = NOPASSWD: /sbin/shutdown” in /etc/sudoers to allow oneadmin to use sudo without password on /sbin/shutdown. If other user is used to run the Green Cloud Scheduler, replace oneadmin with the proper username.

GREEN CLOUD SCHEDULER DEPLOYMENT

1. Install wakeonlan on the control node
2. Optionally but recommended, install cpufreq-utils on the OpenNebula nodes
3. Setup passwordless shutdown for OpenNebula Nodes
   - Execute sudo chmod u+s /sbin/shutdown
   - Add "oneadmin ALL = NOPASSWD: /sbin/shutdown" in /etc/sudoers to allow oneadmin to execute sudo /sbin/shutdown without password. If you run the OpenNebula client from other user than oneadmin, replace oneadmin with your user in the previous command.
4. Edit the /etc/one/oned.conf file to enter the vm creation and deletion hooks. Restart the one deamon.
5. KILL the Default OpenNebula Scheduler process: sudo killall mm_sched
6. Install Ant on the OpenNebula controller node.Enter the java JDK location in the ./build.xml file by changing the jdk.home property. Issue ant in the GlobalLoopController directory to compile the software.
7. Run the GlobalLoopController.jar from the directory where it is located : java –jar ./GlobalLoopController.jar or use ant ant run

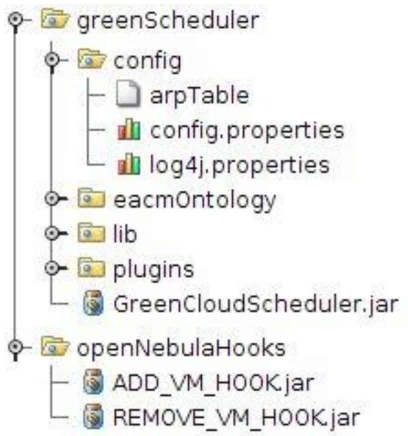

```
greenScheduler
├── config
│   ├── arpTable
│   ├── config.properties
│   └── log4j.properties
├── eacmOntology
├── lib
├── plugins
└── GreenCloudScheduler.jar
openNebulaHooks
├── ADD_VM_HOOK.jar
└── REMOVE_VM_HOOK.jar
```

FIGURE 3: STRUCTURE OF THE DISTRIBUTION DIRECTORY ./DIST .

CONFIGURATION OPTIONS

The next table contains the basic configuration options which need to be addresses when deploying the Green Cloud Scheduler. The configuration options are set from the /config/config.properties file.

Green Cloud Scheduler configuration	Explanation
openNebulaRCPAPIAddress = http://localhost:2633/RPC2	Address and port of the OpenNebula XML RCP API
openNebulaCredentials = oneadmin:oneadmin	Username and password for accessing the OpenNebula API
resourceLoadOptimalValue = 80	Optimal load for server's computing resources in percentage %
resourceLoadVariation = 20	Accept load variation from the defined optimal load value in %
PING_LOCATION = /bin/ping	Path to the ping executable
ARP_LOCATION = /usr/sbin/arp	Path to the arp executable
nodesWakeUpMechanism = wakeonlan | wol	Path of the wake-up mechanism executable
vmMigrationMechanism = live | offline	Live or offline virtual machines migration method
migrationIsEnabled	Disables the VM migration. The GCL will only consider deployment of VMs and turning on/off servers in this case.