ECO$_2$Clouds

# Experimental Awareness of $CO_2$ In Federated Cloud Sourcing

## FP7 – 318048

## How to adopt ECO2Clouds:

*Adaptation of ECO$_2$Clouds to an Infrastructure other than BonFIRE*

**Lead editor:** David García Pérez (ATOS)
**Contributors**: Usman Wajid (UNIMAN), Oliver Barreto (ATOS)
**Version / date**: v1.0 30/10/2014

**Distribution level:** Public

**Project thematic priority**: ICT-2011.1.6 c) Fire Experimentation
**Project start date**: October 1, 2012
**Project duration**: 24 months


This project is partially funded by the European Commission under the 7th Framework Programme - Grant agreement no. 318048

# Table of Contents



# Figures

# 1 Adaptation of ECO2Clouds to different Cloud Infrastructures

Cloud computing is becoming main stream as more and more large data centers are being built to provide online and on-demand services all over the world. At the same time, the amount of energy consumed and Carbon footprint that the Internet economy generates also increases exponentially.


**ECO2Clouds ambition is to provide the means to tackle this problem and create greener cloud environments and enable technology to be more sustainable.**

The ECO2Clouds approach aims to develop a series of techniques to be able to deploy and manage applications over a Cloud Infrastructure (Infrastructure as a Service – IaaS) to minimize the $CO_2$ footprint.

The initial idea of ECO2Clouds project was to create a Proof of Concept (PoC) using the BonFIRE Cloud [1] facility and perform experimentation to demonstrate and validate the rationale behind different ideas coming out of the project. However, efforts were made to design the ECO2Clouds technology in a way that allows it to be used outside the scope of BonFIRE. In order to facilitate how ECO2Clouds technology could be adopted by third parties, the ECO2Clouds Team has created this document which provides an overview of necessary ingredients or core components that make up the ECO2Clouds solution. In order to exploit the potential of ECO2Clouds solutions in any other IaaS platform than BonFIRE, necessary modifications will be required in these components based on the underlying characteristics or the specific nature of the platform

ECO2Clouds technology is built over Open Nebula [6] cloud management stack, and therefore can be easily used to extend or adapt data centers that use this cloud management stack. In this respect, our aim is to facilitate data centers in becoming greener and more environmentally friendly by using ECO2Clouds technology.


If you care about the environmental impact of your cloud infrastructures and you want to start acting, this document helps you find what you need to

adopt ECO2Clouds technology and help the world become greener.

The document is structured as follows:

- Section 2 makes an introduction to ECO2Clouds architecture;
- Section 3 explains the different adaptations necessary to the Accounting Service component which plays a vital role as a service component of the Scheduler, being responsible for directly interacting with the underlying cloud infrastructure
- Section 4 explains the required steps needed in order to a) collect necessary data and metrics from the underlying cloud and b) to manage underlying virtualized infrastructure.
- Section 5 explains the different modifications of the actual ECO2Clouds components.

# 2 ECO2Clouds Architecture

Figure 1 presents the current architecture of ECO2Clouds. The green parts indicate new ECO2Clouds components developed specifically for the project while the orange components indicate original BonFIRE components that are being used in the ECO2Clouds project.

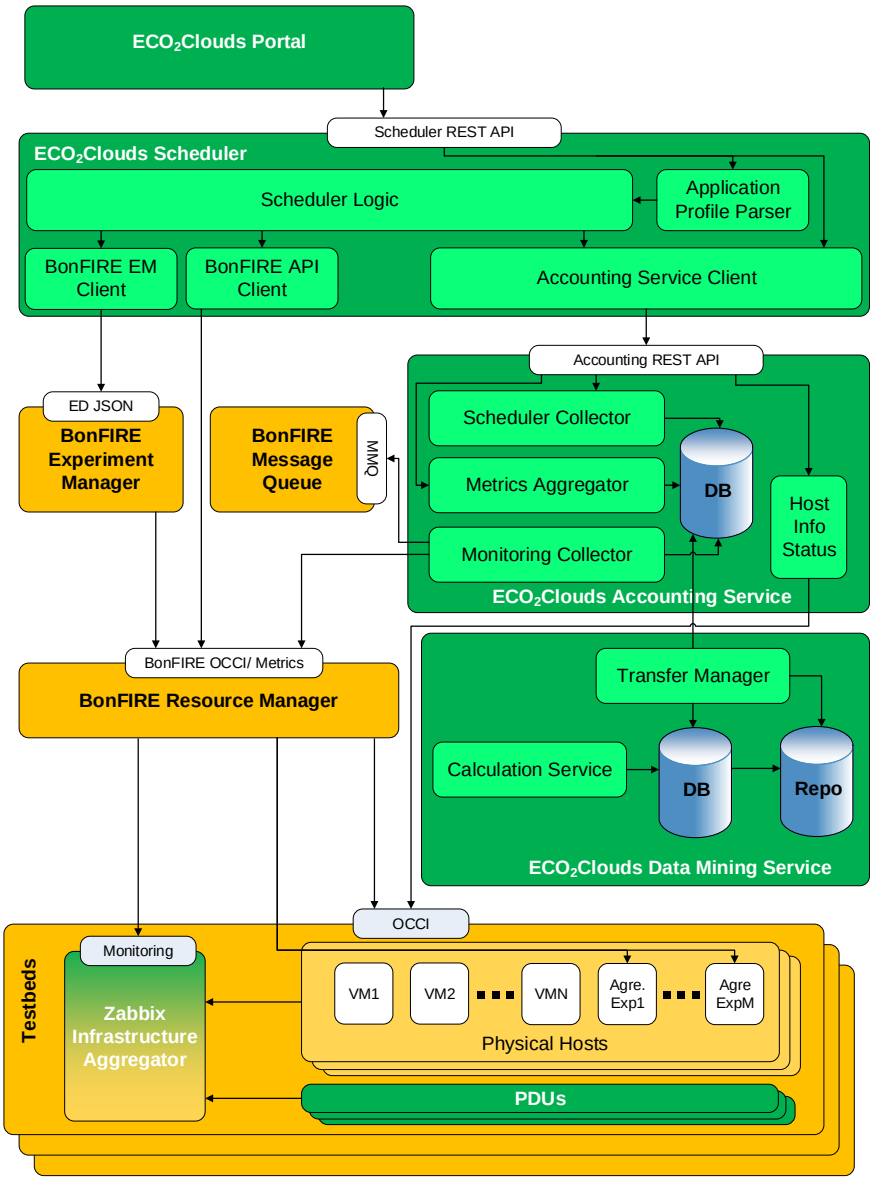



Figure 1: ECO2Clouds Architecture.

The figure makes it clear to visualise the different APIs that make calls to BonFIRE components. These APIs will need to be updated for any adaptation of ECO2Cloud solutions. However, the figure does not make it easy to see what needs to be changed at testbed level.

Also it is necessary to clarify that, as in many other aspects of Cloud Computing, in this case, we understand that an application is a collection of virtual machines

(VM) that fulfil specific set of tasks.

## 2.1 Description of the components

This subsection presents a short description of the key components that make up the ECO2Clouds solution. In order to adapt ECO2Clouds solution for use in any other IaaS platform necessary changes will need to be made in some or all of these components:

- **ECO2Clouds Portal –** This component it is the graphical interface to the final ECO2Clouds user, since it only interacts with the ECO2Clouds Scheduler it may not require any change.
- **ECO2Clouds Scheduler** – This component is responsible to determine green and energy efficient deployment configurations for cloud applications. The Scheduler interacts on one side with the ECO2Clouds Accounting service and on the other side with the BonFIRE Experiment Manager and BonFIRE Resource Manager. Section 5 will highlight possible changes or adaptation actions that may be required to use the Scheduler in other Cloud or IaaS platforms.
- **ECO2Clouds Accounting Service** – The Accounting Service is responsible for collect all monitoring metrics for physical and virtual resources and making them available for the Scheduler so it can perform its optimizations operations. It also logs the different actions of the Scheduler in an SQL database.

  For this component, it is best to detail the necessary changes by subcomponent.

  - Scheduler Collector, Metrics Aggregator, and DB – Those components are not dependent directly of the specific infrastructure, they only interact with other ECO2Clouds components so no changes are necessary.
  - Host Info Status – This component reads the information from OpenNebula (the IaaS Cloud Manager used in BonFIRE), this component needs to be updated in case OpenNebula is not the Cloud Manager at the IaaS infrastructure. More details about this are given in the Section 3.
  - Monitoring Collector – This component depends on two BonFIRE components, the BonFIRE Message Queue and the BonFIRE Resource Managerm to be able to know which resources should be monitored and to collect the different metrics for the selected resources. Section 3 will explain the necessary changes to this component in details.
- **ECO2Clouds Data Mining Service** – Responsible to keep the all the monitoring data generated by ECO2Clouds for later study, it is completely independent of the IaaS provider, so no change is necessary to it.
- **Testbeds** – Basically the IaaS provider. All the changes necessary here are going to be detailed in the Section 3 of the document.

# 3 ECO2Clouds Accounting Service

The ECO2Clouds Accounting Service plays a vital role in realising the potential of ECO2Cloud solution and is the most complex piece to adapt since it directly interacts with the underlying cloud infrastructure in order to gather the monitoring data concerning the infrastructure and the applications running on it.

The monitoring information handled by the Accounting Service is used by Scheduler in its ECO-aware decision making model. In ECO2Clouds the Accounting Service is installed on a VM and provides a REST interface to enable easy access by Scheduler and other modules in the system. Two of its key components are:

- Host Info Status
- Monitoring Collector

### 3.1.1 Host Info Status

This component is responsible for exposing in some way information about the status of the physical hosts in the cloud infrastructure. The HostInfoStatus component will need to be adapted in order to enable the collection of host level information by any other component or API. Further details are discussed in Section 4.1.4.

### 3.1.2 Monitoring Collector

The ECO2Clouds Solution requires the collection of numerous metrics from cloud infrastructures. The metrics are used by real time components (like the ECO2Clouds Scheduler) to make on the fly decisions on optimal VM placement and by offline components (like Data Mining) which can apply several statistical tests in order to find correlations and trends.

The ECO2Clouds Monitoring Collector is responsible for gathering all applicable metrics and status updates from various cloud infrastructures. The metrics and events are stored, in chronological order, in the Accounting DB. In order to request metrics and cloud status updates, the Monitoring Collector uses the ECO2Clouds Metrics Abstraction REST API. The REST API hides the different implementation technologies of various cloud infrastructures. As long as a cloud provider supports the REST API, the Monitoring Collector is able to collect metrics. The following subsections discuss the workflow and implementation of the Monitoring Collector.

#### 3.1.2.1 Workflow of the ECO2Clouds Monitoring Collector

The Monitoring Collector collects metrics from the cloud infrastructure and also cloud status updates. In order to collect metrics from the cloud infrastructure, the Monitoring Collector is using the ECO2Clouds Metrics Abstraction REST API which abstracts cloud components to Site, Physical Host and VM level. Thus, the Monitoring Collector can collect metrics from simple cloud installations which use a single site and a minimal set of physical resources up to multi-site installations with hundreds of physical hosts and thousands of VMs. Each metric for each level is stored in the Accounting DB and can be used by the Scheduler,

other ECO2Clouds components or for offline analysis.

In order to collect cloud status updates, the Monitoring Collector uses a different REST API which abstracts the creation, deletion and migration of VMs. Each action (or event) is analysed and if it passes all consistency tests it is stored in the Accounting DB. The ECO2Clouds Scheduler or other ECO2Clouds components query the Accounting DB in order to retrieve and process the event.

The current implementation integrates with the BonFIRE AMQP Message Queue. The Monitoring Collector can be adapted to plug into the Message Queue of any cloud management software/platform or a new client should be written to access to this kind of information based on the underlying characteristics of the Cloud Manager.

#### 3.1.2.2 Monitoring Collector Deployment

In the ECO2Clouds prototype deployment, the Monitor Collector is deployed on the same VM as the Accounting Service and the Accounting DB. In general, the Monitoring Collector needs read and write access to the Accounting Database, and could in principle be deployed on its own VM or physical host.

# 4 Required Adaptations for Underlying Third Party Clouds (IaaS)

An IaaS Provider that wants to use ECO2Clouds solutions in its infrastructure needs to provide on one side a set of APIs for the ECO2Clouds scheduler be able to deploy, start, stop, resume or delete VMs and on the other side monitoring information about the physical and virtual resources. The following subsections tackle into detail each one of those two aspects:

## 4.1 Monitoring

In order to benefit from ECO2Clouds technology, the following developments and adaptations are necessary to be able to collect all the necessary monitoring information – used by ECO2Clouds solution in making smart and greener decisions.

The following subsections detail the different information necessary for ECO2Clouds to be able to optimize the deployment of an application in a Cloud provider.

### 4.1.1 Energy mix

To be able to optimize the deployment of an application based on $CO_2$ footprint, it is necessary to know the amount of $CO_2$ produced by the energy that a testbed/site is consuming at a given time.

Carbon emissions can be assessed in different ways. Some countries publish the real time energy mix via public web sites. For example, France energy mix can be retrieved through the information service eCO2mix available on the RTE website[1] and data about the energy generation in UK are available through the Balancing Mechanism Reporting System (BMRS) website[2]. The availability of real time data allows us to have a more accurate measurement of carbon emissions. Since the web sites publish the electricity power generated by the different energy sources, it is possible to calculate the emission factor as a weighted average of the sources emission factors on the basis of the percentage of power generated. Formally, if $TP$ is the total power generated in a country, $SP_k$ is the power generated by the k-th source and $ef_k$ is the emission factor related to the k-th source, the total emission factor of the Cloud site can be calculated as follows:

$$ef = \sum_{k=1}^{K} \frac{SP_k}{TP} \cdot ef_k$$

Besides real time values, there are electricity operators that periodically publish aggregated emission factors of the different countries in a specific period. In this case, assuming that we know the average power consumption (AP) for a specific site, the energy (kWh) consumed in a specific period can be estimated by

[1] http://www.rte-france.com/fr/

[2] http://www.bmreports.com

multiplying AP by the number of hours in the considered period. $CO_2$ emissions result by multiplying the energy consumed by the emission factor (that is a constant calculated considering the average energy mix in a certain period).

#### 4.1.1.1 Exposing the energy mix to ECO2Clouds

ECO2Clouds Accounting Service is responsible for gathering all monitoring information from the IaaS layer and providing it to the Scheduler. For ECO2Clouds solution to work, each testbed/cloud-site is required to expose energy mix information (from energy provider) using the REST API. The ECO2Clouds Accounting Service accesses the energy mix information using the following:

```
GET /locations/fr-inria/locationmetrics/aliases

<?xml version="1.0" encoding="UTF-8"?>
<collection href="/locations/fr-inria/locationmetrics/aliases"
xmlns="http://api.bonfire-project.eu/doc/schemas/occi"
type="application/vnd.bonfire+xml">
    <items>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Biomass"
name="Biomass" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/CCGT"
name="CCGT" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Co2"
name="Co2 per kWh" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Coal"
name="Coal" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-
inria/locationmetrics/aliases/Cogeneration" name="Cogeneration"
type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-
inria/locationmetrics/aliases/Exported" name="Exported"
type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Fossil"
name="Fossil" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Gas"
name="Gas" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-
inria/locationmetrics/aliases/Geothermal" name="Geothermal"
type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-
inria/locationmetrics/aliases/Hydraulic" name="Hydraulic"
type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-
inria/locationmetrics/aliases/Imported" name="Imported"
type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-
inria/locationmetrics/aliases/NPSHydro" name="NPS hydro"
type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Nuclear"
name="Nuclear" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/OCGT"
name="OCGT" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Oil"
name="Oil" type="application/vnd.bonfire+xml"/>
```

```
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Other"
name="Other" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-
inria/locationmetrics/aliases/PumpedStorage" name="Pumped storage"
type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Solar"
name="Solar" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-
inria/locationmetrics/aliases/GridTotal" name="Grid Total"
type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-
inria/locationmetrics/aliases/TotalGreen" name="Total green"
type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Water"
name="Water" type="application/vnd.bonfire+xml"/>
      <locationmetric href="/locations/fr-inria/locationmetrics/aliases/Wind"
name="Wind" type="application/vnd.bonfire+xml"/>
    </items>
</collection>
```

The URL: /locations/fr-inria/locationmetrics/aliases is BonFIRE specific, following the format /locations/<testbed-name>/ locationmetrics/aliases . This format can be kept or the REST client at the Monitoring Collector should be updated for any adaptation of the Accounting Service.

For each one of the types of energy source the metric just indicates the % for that source:

```
<?xml version="1.0" encoding="UTF-8"?>
<metric>
  <itemid>103437</itemid>
  <value>1.300000</value>
  <clock>1402584837</clock>
  <name>Wind</name>
  <key>Wind</key>
  <name>Wind</name>
  <unit>%</unit>
  <link rel="history" href="/locations/fr-
inria/locationmetrics/aliases/Wind/history_metrics"
type="application/vnd.bonfire+xml"/>
</metric>
```

While $CO_2$ indicates the grams of $CO_2$ produced by kw:

```
<?xml version="1.0" encoding="UTF-8"?>
<metric>
  <itemid>103437</itemid>
  <value>461.281824</value>
  <clock>1412865287</clock>
  <name>Wind</name>
  <key>Co2</key>
  <name>Co2</name>
  <unit>g/kWh</unit>
  <link rel="history" href="/locations/fr-
inria/locationmetrics/aliases/Co2/history_metrics"
```

```
type="application/vnd.bonfire+xml"/>
</metric>
```

### 4.1.2 Physical Data Center Level: PDUs

It is necessary to monitor the energy usage of the different physical resources in the IaaS facility. To do so, managed power distribution units (PDUs) can provide sufficiently accurate data. This needs to include power used by cooling equipment, compute, networking and storage elements used to run the infrastructure and those used directly by the clients of the infrastructure. Ideally, power distribution units would be able to report on energy usage for each plug it monitors with sufficient precision to report new values every minute or more frequently. Unfortunately, we have found that some of the off the shelf PDUs do not have enough precision when reporting on energy usage through a given plug. Therefore, we recommend to measure frequently only the power usage of each plug and derive the energy usage from such values if necessary.

Moreover, another issue to address when using PDUs is that computing elements are not always directly linked to power plugs. Blade servers for example might share 2 power plugs between 10 machines. We have found it important to build a monitoring infrastructure where power usage of each node is collected. When Intelligent Platform Management Interface (IPMI) enables self-monitoring of power usage by each node, power consumption assessment can be done dynamically. The power consumption of a node $k$ can be calculated as:

$$PowerConsumption_k = PowerConsumptionPlug \cdot \frac{PowerConsumptionIPMI_k}{\sum_k PowerConsumptionIPMI_k}$$

where $PowerConsumptionPlug$ is the power measured at plug level and $PowerConsumptionIPMI_k$ is the IPMI measurement of power consumption of the k-th node.

If chilling equipment cannot be measured, an estimation of the PUE of the datacentre room used can be enough.

### 4.1.3 Metrics and Monitoring Infrastructure

In ECO2Clouds the quantification of energy consumption and environmental impact is performed based on the availability of a layered set of metrics. In fact, the model adopted in ECO2Clouds consists of (i) an *infrastructure layer* that refers to a IaaS provider or several and their resources; (ii) a *virtualization layer* that contains the Virtual Machines (VMs), which are responsible to host the applications; (iii) the *application layer* that includes the applications running on one or more VMs.

For the IaaS provider needs to focus in the two first layers: infrastructure and virtualization. For each one of those layers, the provider needs to enable the following metrics:

Virtual Machine metrics (virtualization layer):

| Metric | Definition | Unit |
|---|---|---|
| **CPU usage** | Current processor utilization percentage (inside a VM) for a running virtual machine. It is calculated as the ratio between the amount of used CPU and the amount of allocated CPU. | % |
| **Storage usage** | Storage utilization percentage on the corresponding storage device, computed as the ratio between the used disk space and the allocated disk space. | % |
| **I/O usage** | Percentage of process execution time in which the disk is busy with read/write activity. | % |
| **Memory usage** | Ratio of the size of the portion of memory used by the VM to the total amount of memory available. | % |
| **Power consumption** | The power currently consumed by the analysed VM. This is determined by VM Free Memory, Number of VMs running on a host, Processor Load and Total Memory | W |
| **Disk IOPS** | This metric aims to assess the I / O operations of a virtual resource. This metric only considers storage operations. This is determined with the help of VM energy consumption, Incoming traffic and Outgoing traffic | Ops/s |

The VM Power Consumption metrics relies on the host energy consumption and uses CPU utilization as major indicator. However, the infrastructure provider can consider various ways (from existing literature) of calculating this metrics.

Physical Hosts (infrastructure layer):

| Metric | Definition | Unit |
|---|---|---|
| **Power consumption** | The power consumed by the analysed host in a specific time period. | W |
| **Disk IOPS** | This metric checks the I/O operations of the disk within a host. | IOPS/s |
| **CPU utilization** | The average utilization of the processors inside a host. For each processor, this metric indicates how much of the processor capacity is used. | % |
| **Availability** | Checks the availability of a host. | Boolean |

Site (infrastructure layer):

| Metric | Definition | Unit |
|---|---|---|
| **Site utilization** | Current utilisation of a single site. The metric is defined by (available cores) / (total cores). | % |
| **Storage utilization** | Percentage of the frontend storage used provides a good estimation of the site utilization. If necessary, this metric can be implemented for single worker hosts as well. | % |
| **Availability** | A site will be available, if the OpenNebula OCCI server provides a reasonable answer to a request. Furthermore, at least one host has to be seen "online" (with one available core) by OpenNebula.<br>It checks if the available count of hosts multiplied with the tcp port 8443 (OCCI server reachable) and the tcp port 4567 (OCCI server listening) is equal or higher than 1. If yes, at least one host is "online" and the availability is true. | Boolean |
| **PUE** | Metric used to determine the energy efficiency of a site. It is measured as the ratio between the total facility power and the power that is used by the computing equipment. This metric is more or less represented by a static value depending on the conditions of each data center. For instance, at HLRS a fixed value is necessary due to the static energy mix defined in the power supplier contract. | None |

#### 4.1.3.1 Energy estimation for a VM

Although we indicated that the IaaS provider can select their own way to calculate the energy consumed by a VM, here we propose the solution selected in the ECO2Clouds project.

The assessment of the *Energy Consumption* is performed at the host level for the physical nodes. The measurement at host level is enabled through the installation of power distribution units (PDUs). The monitoring system can collect data regarding the consumed power and energy for each physical node in VA, W and Wh. These measurements are the base for the calculation of the power consumption per VM.

In fact, at the virtualization layer, the power consumption of a VM is estimated as follows:

$$\frac{IDLE\ power\ of\ the\ host}{Number\ of\ VMs} + Dynamic\ power\ of\ the\ host \times \frac{CPU\ used\ by\ the\ VM}{CPU\ used\ by\ all\ VMs\ running\ on\ the\ host}$$

All these values, measured at the infrastructure layer are made available on the user data plane. Therefore, both the infrastructure provider and the users are able to keep track of the power consumption of a VM.

#### 4.1.3.2 Exposing the metrics to ECO2Clouds

Even though ECO2Clouds used Zabbix[3] during its proof of concept, the access to the monitoring information from the ECO2Clouds Accounting Service was abstracted from the Zabbix monitoring solution, enabling an IaaS provider to use their favourite or already installed monitoring solution and only building the compatible interface around it.

Similar to the energy mix REST service (see subsection 4.1.1.1), the ECO2Clouds Monitoring Collector is expecting a REST service with the following structure. To get the list of metrics for a site:

```
GET /locations/fr-inria/hostmetrics
```

```
<?xml version="1.0" encoding="UTF-8"?>
<availablehostmetrics>
  <procnum>
    <key>proc.num[]</key>
    <enabled>true</enabled>
    <unit>proc</unit>
    <name>Number of processes</name>
  </procnum>
  <cpuload>
    <key>system.cpu.load[,avg1]</key>
    <enabled>true</enabled>
    <unit>%</unit>
    <name>Processor load</name>
  </cpuload>
  <cpuutil>
    <key>system.cpu.util[,user,avg1]</key>
    <enabled>true</enabled>
    <unit>%</unit>
    <name>CPU user time (avg1)</name>
  </cpuutil>
  <memfree>
    <key>vm.memory.size[free]</key>
    <enabled>true</enabled>
    <unit>MB</unit>
    <name>Free memory</name>
  </memfree>
  <memtotal>
    <key>vm.memory.size[total]</key>
    <enabled>true</enabled>
    <unit>MB</unit>
    <name>Total memory</name>
  </memtotal>
  <swapfree>
    <key>system.swap.size[,free]</key>
    <enabled>true</enabled>
    <unit>B</unit>
```

[3] http://www.zabbix.com/

```
    <name>Free swap space</name>
  </swapfree>
  <runningvm>
    <key>custom.vms.running</key>
    <enabled>true</enabled>
    <unit>Vm</unit>
    <name>Number of VMs running</name>
  </runningvm>
  <co2g>
    <key>power.co2.generated</key>
    <enabled>true</enabled>
    <unit>g</unit>
    <name>CO2 generation per 30s</name>
  </co2g>
  <conswh>
    <key>power.currentwh.real</key>
    <enabled>true</enabled>
    <unit>Wh</unit>
    <name>Aggregate energy usage</name>
  </conswh>
  <consva>
    <key>power.currentva.real</key>
    <enabled>true</enabled>
    <unit>VA</unit>
    <name>Apparent power</name>
  </consva>
  <consw>
    <key>power.currentw.real</key>
    <enabled>true</enabled>
    <unit>W</unit>
    <name>Real power</name>
  </consw>
  <freespacesrv>
    <key>vfs.fs.size[/srv,free]</key>
    <enabled>true</enabled>
    <unit>B</unit>
    <name>Free space on /srv</name>
  </freespacesrv>
  <Availability>
    <key>one.availability</key>
    <enabled>true</enabled>
    <unit></unit>
    <name>Availability</name>
  </Availability>
  <IOPS>
    <key>custom.vfs.iops</key>
    <enabled>true</enabled>
    <unit></unit>
    <name>Disk IOPS</name>
  </IOPS>
  <cpuUtilization>
    <key>custom.cpu.utilization</key>
    <enabled>true</enabled>
    <unit>%</unit>
    <name>CPU utilization</name>
  </cpuUtilization>
  <PowerConsumption>
    <key>power.consumption</key>
    <enabled>true</enabled>
    <unit>W</unit>
    <name>Power consumption</name>
  </PowerConsumption>
</availablehostmetrics>
```

For each of those metrics a description was given in the previous section.

The Monitoring Collector service finds how many hosts are available at each IaaS Provider by a REST query with the following structure:

```
GET /locations/fr-inria/hostmetrics
```

```
<?xml version="1.0" encoding="UTF-8"?>
<collection xmlns="http://api.bonfire-project.eu/doc/schemas/occi"
href="/locations/fr-inria/hosts">
  <link rel="common_metrics" href="/locations/fr-inria/hostmetrics/aliases"
type="application/vnd.bonfire+xml"/>
  <items>
    <host href="locations/fr-inria/hosts/bonfire-blade-1"
type="application/vnd.bonfire+xml" name="bonfire-blade-1">
      <name>bonfire-blade-1</name>
    </host>
    <host href="locations/fr-inria/hosts/bonfire-blade-2"
type="application/vnd.bonfire+xml" name="bonfire-blade-2">
      <name>bonfire-blade-2</name>
    </host>
    <host href="locations/fr-inria/hosts/bonfire-blade-3"
type="application/vnd.bonfire+xml" name="bonfire-blade-3">
      <name>bonfire-blade-3</name>
    </host>
    <host href="locations/fr-inria/hosts/bonfire-blade-4"
type="application/vnd.bonfire+xml" name="bonfire-blade-4">
      <name>bonfire-blade-4</name>
    </host>
  </items>
</collection>
```

And finally, to know the value of a metric for a specific host

```
GET /locations/fr-inria/hosts/bonfire-blade-1/hostmetrics/aliases/PowerConsumption
```

```
<?xml version="1.0" encoding="UTF-8"?>
<metric>
  <itemid>107054</itemid>
  <value>140</value>
  <clock>1412872126</clock>
  <name>Power consumption</name>
  <key>PowerConsumption</key>
  <key_>power.consumption</key_>
  <unit>W</unit>
</metric>
```

Again, the structure or the REST paths can be changed, but this will imply to change the REST client inside the ECO2Clouds Monitoring Collector.

For the VMs the process is similar. To get the different metrics for a virtual machine:

```
GET /locations/fr-inria/computes/67072/vmmetrics/aliases
```

```
<?xml version="1.0" encoding="UTF-8"?>
<metric>
  <procnum>
    <key>proc.num[]</key>
    <enabled>true</enabled>
    <unit>proc</unit>
    <name># of procs</name>
  </procnum>
  <cpuload>
    <key>system.cpu.load[,avg1]</key>
    <enabled>true</enabled>
```

```
    <unit>%</unit>
    <name>CPU load</name>
  </cpuload>
  <cpuutil>
    <key>system.cpu.util[,user,avg1]</key>
    <enabled>true</enabled>
    <unit>%</unit>
    <name>CPU usage</name>
  </cpuutil>
  <ioutil>
    <key>system.cpu.util[,iowait,avg1]</key>
    <enabled>true</enabled>
    <unit>%</unit>
    <name>I/O usage</name>
  </ioutil>
  <memused>
    <key>vm.memory.usage</key>
    <enabled>true</enabled>
    <unit>%</unit>
    <name>Memory usage</name>
  </memused>
  <memfree>
    <key>vm.memory.size[free]</key>
    <enabled>true</enabled>
    <unit>MB</unit>
    <name>Free memory</name>
  </memfree>
  <memtotal>
    <key>vm.memory.size[total]</key>
    <enabled>true</enabled>
    <unit>MB</unit>
    <name>Total memory</name>
  </memtotal>
  <swapfree>
    <key>system.swap.size[,free]</key>
    <enabled>true</enabled>
    <unit>B</unit>
    <name>Free swap</name>
  </swapfree>
  <swaptotal>
    <key>system.swap.size[,total]</key>
    <enabled>true</enabled>
    <unit>B</unit>
    <name>Total swap</name>
  </swaptotal>
  <netifin>
    <key>net.if.in[eth0,bytes]</key>
    <enabled>true</enabled>
    <unit>KBps</unit>
    <name>Bytes in</name>
  </netifin>
  <netifout>
    <key>net.if.out[eth0,bytes]</key>
    <enabled>true</enabled>
    <unit>KBps</unit>
    <name>Bytes out</name>
  </netifout>
  <diskfree>
    <key>vfs.fs.size[/,free]</key>
    <enabled>true</enabled>
    <unit>B</unit>
    <name>Free storage</name>
  </diskfree>
  <disktotal>
    <key>vfs.fs.size[/,total]</key>
```

```
    <enabled>true</enabled>
    <unit>B</unit>
    <name>Total storage</name>
  </disktotal>
  <diskusage>
    <key>vfs.fs.size[/,pused]</key>
    <enabled>true</enabled>
    <unit>%</unit>
    <name>Storage usage</name>
  </diskusage>
  <power>
    <key>energy.vm.total</key>
    <enabled>true</enabled>
    <unit>W</unit>
    <name>Power consumption</name>
  </power>
  <iops>
    <key>custom.vfs.iops</key>
    <enabled>true</enabled>
    <unit>IOPS</unit>
    <name>IOPS</name>
  </iops>
</metric>
```

And similar to the hosts, to get the last value of a metric for a VM:

```
GET /locations/fr-inria/computes/67072/vmmetrics/aliases/power

<?xml version="1.0" encoding="UTF-8"?>
<metric>
  <itemid>23169</itemid>
  <value>38.100483</value>
  <clock>1412873379</clock>
  <name>power</name>
  <key>power</key>
  <key_>energy.vm.total</key_>
  <unit>W</unit>
</metric>
```

### 4.1.4 Cloud Infrastructure Monitoring (Physical and Virtualized Hosts)

Finally, in ECO2Clouds the Open Nebula based cloud management software exposes the following information about each host that is used by the Scheduler in its application deployment decision making model:

```
ID  NAME                RVM   TCPU   FCPU   ACPU   TMEM   FMEM   AMEM STAT
552 bonfire-blade-1     15    2400   2369   50     63.7G  40.1G  29.4G on
450 bonfire-blade-2     11    2400   2243   350    63.7G  31.1G  31.4G on
451 bonfire-blade-3     13    2400   2296   650    63.7G  38.1G  38.4G on
448 bonfire-blade-4     16    2400   2375   0      63.7G  35.3G  35.7G on
```

Where each of those fields mean:

- RVM – Total number of running VMs in the physical host.
- TCPU – Total CPU available in the host (percentage)
- FCPU - Free CPU in the host (percentage)
- ACPU – Available CPU (that can be allocated to create VMs - percentage)
- TMEN – Total memory

- FMEN – Free memory
- STAT – Status of the host (on/off)

The ECO2Clouds Host Status Information accesses to this OpenNebula information. This kind of information it is usually provided by other Cloud Manager such as VMWare or OpenStack. The ECO2Clouds Host Status Information module should be rewritten to access to that specific API.

## 4.2 IaaS APIs

ECO2Clouds interacts with the IaaS provider in two ways, one to create, manage or delete resources, in this case VMs, and to know of the status of the different VMs in the infrastructure. To do so, ECO2Clouds employs the BonFIRE API and it subscribe to the BonFIRE Management MQ.

### 4.2.1 Cloud Manager API

The IaaS manager should provide an API to allow the creating of VMs in a specific host. Basically, this API should allow ECO2Clouds Scheduler to bypass the Scheduler inside the Cloud Manager software.

This can be done quite easily with OpenNebula Cloud Manager and we believe it should not be difficult to implement in any of the other open source Cloud Managers such as OpenStack.

### 4.2.2 Message Queue

The ECO2Clouds Monitoring Collector needs to know when a new VM is created or deleted in the infrastructure to know when to start or stop collecting monitoring information for it.

In the current ECO2Clouds proof of concept implemented over BonFIRE the Monitoring Collector makes use of the BonFIRE Message Queue.

BonFIRE has a message queue to which different services can subscribe to know when a resource has been created, deleted or its state modify. The message follows this format:

```
{
        "timestamp":1373881015,
        "eventType":"create",
        "objectType":"compute",
        "objectId":"/computes/257",
        "source":"res-mng",
        "groupId":"eco2clouds",
        "userId":"eco2clouds"
}
```

For the virtual machines the following states are monitored: created, stopped, done (deleted).

Similar MMQ service should be provided by the IaaS layer, if necessary, the Message Queue client at the ECO2Clouds Monitoring Collector should be also modified.

# 5 Other ECO2Clouds Components Modifications

As seen in the architecture, ECO2Clouds is composed of three major components:

- **ECO2Clouds Portal**
- **ECO2Clouds Scheduler**

This section describes all the necessary changes needed for each of these components in order to use ECO2Clouds.

## 5.1 ECO2Clouds Portal

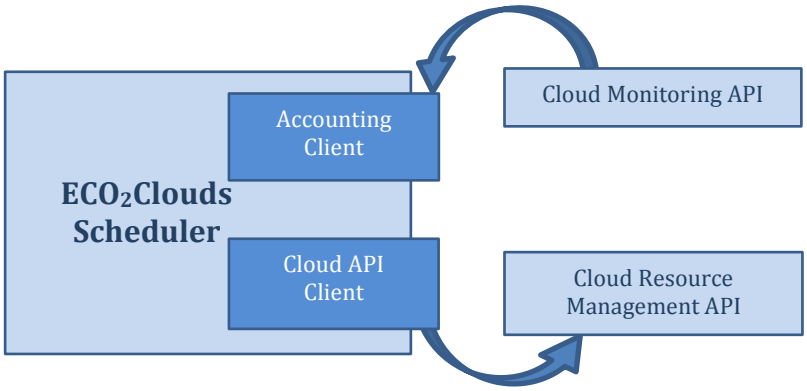
The ECO2Clouds Portal provides a visual interface to assist use in setting up experiments on the federated cloud infrastructure (including creation of virtual resources). The portal also enables users to subsequently monitor the performance (e.g. energy consumption and carbon footprint) of running applications. The portal only interacts with the ECO2Clouds Scheduler, so no changes are necessary for this component.

## 5.2 ECO2Clouds Scheduler

ECO2Clouds Scheduler minimizes the environmental impact of cloud computing by employing a set of predefined energy and $CO_2$ aware scheduling strategies. The scheduling strategies proactively consider current utilization ratios, energy consumption and $CO_2$ footprint of cloud resources to identify deployment configurations and make recommendations for cloud applications that can contribute towards minimizing their overall environmental impact.

The ECO2Clouds Scheduler is designed as a REST service that can be hosted on a VM within the cloud environment. It constantly tries to reduce the energy consumption and $CO_2$ footprint of federated clouds while satisfying application level requirements and assuring effective utilization of available resources.

To perform its operations, the ECO2Clouds Scheduler needs constant access to monitoring information (or metrics) and be able create, stop, shutdown, resume, delete VMs in the IaaS with specific contextualization information.




**Accounting Client**

The Scheduler only interacts with the ECO2Clouds Accounting Service, so if necessary changes have been performed at Accounting Service level this interface may not require changes while adapting ECO2Clouds for other Cloud platforms.

**Cloud API Client**

To deploy a new application in a BonFIRE ECO2Clouds makes use of two REST clients:

- BonFIRE Experiment Manager Client – This client submits what it is known as a BonFIRE Experiment Descriptor[4], a JSON document describing a collection of VMs to be deployed together with the relation between them.

  This document should be translated by the new Cloud Manager client, that needs to be implemented, into individual calls to create the different VMs of the application to be deployed or, into whatever document the new IaaS Cloud provider supports to deploy a collection of VMs and storage resources.

  BonFIRE Resource Manager Client – This client interacts with the BonFIRE Resource Manager that allows performing typical REST operations over resources such as VMs, storages, and so on. This client should be rewritten to interact with the API provided by the new Cloud Manager.

The ECO2Clouds Scheduler performs the application deployment decision making at two different levels – akin to the federated cloud infrastructure it was designed for.

**Deployment/Scheduling Decision at Testbed Level:**

While making a deployment decision at testbed level or selecting a testbed from those available in a cloud federation, the suitability of the testbed is determined by a combination of multi-criteria optimization and load balancing techniques.

- *Individual Deployment* performs the selection of a testbed separately for each individual VM in the deployment request. In this mode, the VMs in a single deployment request (representing a distributed application) may be deployed on different testbeds. After the allocation of a single VM, the suitability of a testbed may change and thus, the next VM in the same deployment request may be allocated to another testbed that fits the suitability criteria.
- *Bulk Deployment* performs the selection of a testbed for all VMs in a particular deployment request. Here the VM requirements of all VMs (in a deployment request or application profile) are calculated and the deployment decision is made while considering the aggregated requirements of all VMs. In this respect, all VMs (belonging to an application) will be deployed on a single suitable testbed.

[4] http://doc.bonfire-project.eu/R4.0.5/client-tools/exp-desc/json.html

**Deployment/Scheduling Decision at Physical Host Level:**

After determining a suitable testbed the next step for the ECO2Clouds Scheduler is to determine physical host level deployment configuration of new VMs/applications. Since in ECO2Clouds, the physical hosts in a testbed use the same energy mix, the optimization potential lies in reducing the overall energy consumption of the physical hosts. Hence, at this stage the objective of ECO2Clouds Scheduler is to reduce the overall energy consumption of the testbed, which can directly contribute towards reducing the $CO_2$ emissions. In this respect, two physical host level deployment policies have been introduced:

- *Max-Utilization* or Task Consolidation tries to maximize the utilization of individual physical hosts by deploying VMs on the most energy consuming hosts.
- *Min-Utilization* or Task Dispersal tries to minimize the utilization of individual hosts by deploying VMs on least energy consuming hosts, in order to balance the workload across available hosts.

It is important to consider that in addition to energy consumption, the above strategies consider a delicate mix of resource level parameters e.g. CPU, Memory and number of running VMs to determine suitable hosts for application deployment.

The switch between different deployment strategies (both at testbed and physical host level) can be made by using a simple REST command e.g.:

- *Individual Deployment:* http://localhost:8080/scheduler/boo/idm
- *Bulk Deployment:* http://localhost:8080/scheduler/boo/bdm
- *Max-Utilization:* http://localhost:8080/scheduler/soo/min
- *Min-Utilization:* http://localhost:8080/scheduler/soo/max

In ECO2Clouds we have experimented with the above strategies and found that IDM (testbed level) and Max-Utilization (host level) combination yields best results. However, the experimental results obtained in ECO2Clouds are inherently dependent on the information concerning the type and capacity of the available hosts, their energy consumption characteristics, and the current load of the testbed in BonFIRE. Based on the variations in these factors (in different cloud environments) one strategy may perform better than the other. In this respect, experimentation concerning different strategies of the ECO2Clouds Scheduler in a different cloud environment can help to determine the best practices for the particular cloud environment.

# 6 References

[1] BonFIRE – www.bonfire-project.eu, last visited on 13 of May of 2014.
[2] D3.3: Realisation of an enhanced Monitoring and Data Analysis environment (I). 28/06/2013. ECO2Clouds Project.
[3] D2.4: Updated ECO2Clouds Architecture. 28/02/2014. ECO2Clouds Project.
[4] D3.2: Design of an Efficient Monitoring Environment for Complex IT Infrastructures. 28/03/2013. ECO2Clouds Project.
[5] Zabbix, the enterprise-class Monitoring for everyone. www.zabbix.com, last visited on 13 of May 2014.
[6] OpenNebula – Flexible Enterprise Cloud Made Simple: www.opennebula.org.