Experimental Awareness of CO_2 In Federated Cloud Sourcing

FP7 – 318048

D3.4: Realisation of an enhanced Monitoring and Data Analysis environment (II)

Version / date: 1.2, 09/09/2014

WP3: Energy Metrics, Monitoring, and Application Characterization

Lead editor: HLRS

Distribution level: Restricted

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

Deliverable information sheet:

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

Deliverable date due: M21
Deliverable Submitted: 09.09.2014
Lead organization: HLRS
Contributors: HLRS, POLIMI, INRIA, ATOS, EPCC

Reviewers:
Kostas Kavoussanakis (EPCC)
David Garcia Perez (ATOS)

Version	Date	Contributors	Sections Affected
0.1	25/04/2014	Pavel Skvortsov (HLRS)	ToC, structure
0.2	08/05/2014	David Margery (INRIA)	Section 5
0.3	13/05/2014	Axel Tenschert (HLRS)	Section 4
0.4	15/05/2014	Axel Tenschert (HLRS)	Added energy mix metrics in Section 4
0.5	15/05/2014	Pavel Skvortsov (HLRS)	Section 6
0.5draft	21/05/2014	Cinzia Cappiello, Pierluigi Plebani (POLIMI)	Section 6, data analysis draft
0.6	10/06/2014	Axel Tenschert, Pavel Skvortsov (HLRS)	Executive Summary, Introduction, Status as per D3.3, Update Section 4, Conclusions, References and general format
0.7	17/06/2014	Cinzia Cappiello, Pierluigi Plebani (POLIMI)	Section 4, 6
0.8	18/06/2014	Axel Tenschert, Pavel Skvortsov (HLRS)	Finalization for internal review

0.9	26/06/2014	Axel Tenschert, Pavel Skvortsov, Michael Gienger (HLRS); Cinzia Cappiello, Pierluigi Plebani (POLIMI)	HLRS: 1st iteration of internal review from Kostas Kavoussanakis (EPCC), Section 1, 2, 3, 4, 6 ,7 POLIMI: update of Section 4 and 6
1.0	30/06/2014	Axel Tenschert, Pavel Skvortsov, Michael Gienger (HLRS); Iakovos Panourgias (EPCC); David Margery (INRIA), David Garcia Perez (ATOS)	Added Section 4 "Monitoring Components"
1.1	04/07/2014	Axel Tenschert, Pavel Skvortsov (HLRS); David Garcia Perez (ATOS); Iakovos Panourgias (EPCC); Cinzia Cappiello (POLIMI)	Final update based on 2nd internal review
1.2	09/09/2014	Kostas Kavoussanakis (EPCC); Axel Tenschert (HLRS); David Margery (Inria)	Finalization of updates based on the internal review.

Table of Contents

Figures

Tables

1 Executive Summary

D3.4 is the continuation of the D3.3 “Realisation of an Enhanced Monitoring and Data Analysis Environment” (D3.3 2013) deliverable. D3.4 presents the refined realisation of the monitoring infrastructure of the ECO$_2$Clouds project based on the concept from D3.2 (D3.2 2013) and initial realisation from D3.3. This includes definition and monitoring of metrics, as well as their storage and analysis.

As it was presented in D3.3, metrics were implemented, whose main purpose is to measure the carbon emissions of the ECO$_2$Clouds project infrastructure. For the definition of the set of metrics required for the ECO$_2$Clouds project we adopted a layered metric approach. Thus, metrics were defined and implemented for each of the layers including a set measuring the used energy mix of each provider. The layered approach is based on the assumption that the cloud infrastructure is virtualised and that consequently it is possible to distinguish three different layers: the application layer, the virtual machine (VM) layer and the infrastructure layer. Further, the selected monitoring tool is Zabbix were two types of it are used, Zabbix server and Zabbix agents. The monitoring metrics are implemented as Zabbix items or customized scripts linked and integrated into Zabbix server. The Zabbix server needs to be implemented at each site in order to allow the use of the Zabbix monitoring system for the physical infrastructure. Hence, each provider has a Zabbix server communicating to Zabbix agents installed at each physical node.

In this deliverable, we describe in detail the set of the monitored metrics for each of three levels (Application, VM and Infrastructure), having adopted them to the needs of the ECO$_2$Clouds use cases. Also we list the metrics dropped from the previously defined set, due to redundancy and inefficiency. The common practices and problems with the monitoring metrics are also discussed. We also show the way how providers are able to measure (estimate) the carbon footprint by using the implemented metrics. Furthermore, a brief overview of components beyond WP3 related to the monitoring approach is given in order to define the monitoring metrics data flow from the pure monitoring itself to the availability of measurement results in the Accounting Domain.

Additionally, we describe the architecture, implementation and concept of the enhanced Data Mining approach, which consists of data reduction and data analysis components.

First, we focus on the concrete method of data reduction (aggregation). Data aggregation is performed daily over 10-15 Mb datasets generated per day. Based on the reduction rate read from a config file, we aggregate the historic-data of physical hosts which is not needed for the Scheduler. However, only the data of the finished experiments are considered in order not to affect the running ones. The aggregation and transfer of the daily data set take about 5 minutes each.

Then we provide an explanation about the analysis that we perform on monitored data. Such analysis aims to extract useful knowledge able to support the applications deployment at design and run time. In particular, so far we use correlation analysis of the raw metric data to understand influences among metrics that can be exploited to forecast metric values and the impact of different execution strategies on the application's energy consumption. Furthermore, a statistical analysis of historical values related to the energy mix used in the national grids allowed us to extract patterns useful to support the estimation of the carbon emissions during the execution of applications in the different cloud sites. The results of the correlation analysis are stored at the Accounting database as a separate table and can be used by the Scheduler for optimisation of the application deployment.

2 Introduction

The purpose of the deliverable D3.4 "Realisation of an enhanced Monitoring and Data Analysis environment (II)" is to refine the monitoring metrics implementation and provide more details about the Data Mining approach, which were described in D3.3.

First, the goal is to improve the set of the monitored metrics for each of three levels (Application, VM and Infrastructure), which includes their adaptation to the needs of the use cases. In addition, some of the metrics were derived by the accounting service that calculates "easy" metrics such as, for instance, 1 / (metric). Further, the monitoring metrics data flow is presented in order to describe how the measurement results become available in the Accounting Domain.

Second, the goal of this deliverable is to refine the Data Mining approach: we present in more detail the way data reduction (aggregation) is performed, and focus on the correlation analysis of the raw data.

2.1 Document Organisation

The document is organised as follows:

- In Section 3 we summarize the content of the previous deliverable;
- Section 4 identifies the data flow of the monitoring metrics measurement results;
- Section 5 presents refinement of the previously introduced monitoring metrics;
- In Section 6 we focus on measuring the produced carbon footprint;
- Section 7 describes the refined Data Mining approach including data aggregation and data analysis;

Finally, in Section 8 we conclude this deliverable with a summary.

2.2 Glossary

API	Application Programming Interface
AS	Accounting Service
A-EP	Application Energy Productivity
A-GE	Application Green Efficiency
A-PUE	Application Power Usage Effectiveness
CUE	Carbon Usage Effectiveness
CPU	Central Processing Unit
DB	Database
DCeP	Data Center Energy Productivity
GEC	Green Efficiency Coefficient
I/O	Input / Output
IOPS	Input / Output Operations Per Second
IT	Information Technology
OID	Object Identifier
PDU	Power Distribution Unit
PUE	Power Usage Effectiveness
VM	Virtual Machine
VM-EP	Virtual Machine Energy productivity
VM-GE	Virtual Machine Green Efficiency
VM-PUE	Virtual Machine Power Usage Effectiveness

3 Status as per D3.3

In D3.3 the monitoring metrics were defined based on previous updates. The D3.3 metrics are the base for the final refinements of the monitoring metrics presented in D3.4. Additionally, a first iteration of the monitoring metrics implementations based on the monitoring architecture was finished. Further, the implementations were used by the case studies in order to understand the whole monitoring infrastructure behaviour. This was indicated as a major step for supporting WP4 and WP5 and in general for the following milestones. D3.3 provided the improved monitoring architecture and the first sketch of the data mining approach.

The monitoring architecture is based on a *federated cloud model.* A federated cloud can be constructed as a set of different hardware resources (e.g. hosts, storage, network devices) as presented in D3.3. Further, the monitoring metrics were defined in a hierarchical way related to this *federated cloud model.* The layered monitoring approach is composed of the *infrastructure*, the *virtualization* and the *application* layer. The layers were presented in D3.3 and presented in Figure 1: The Layered Monitoring Approach.

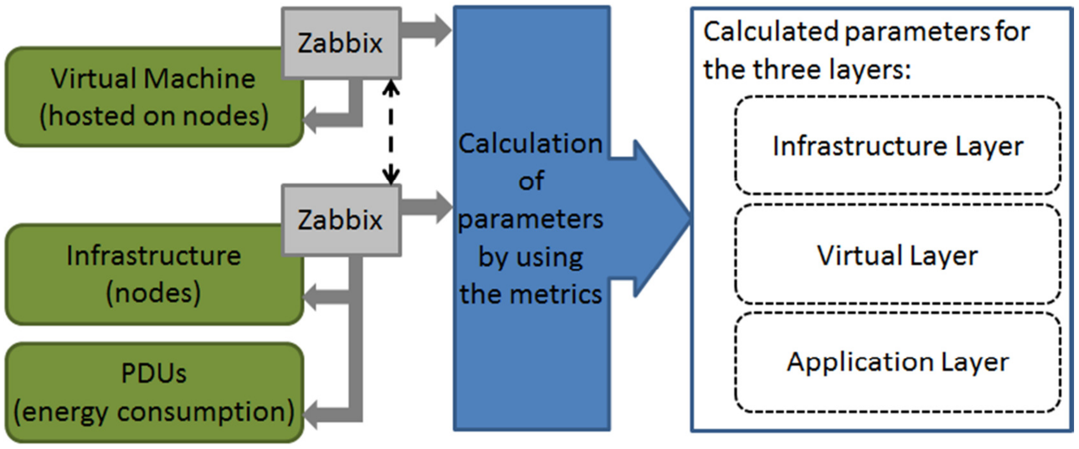

Zabbix
Virtual Machine (hosted on nodes)
Zabbix
Infrastructure (nodes)
PDUs (energy consumption)
Calculation of parameters by using the metrics
Calculated parameters for the three layers:
Infrastructure Layer
Virtual Layer
Application Layer

Figure 1: The Layered Monitoring Approach

It demonstrates the structure of the layered approach by presenting the virtual machines (VMs), the infrastructure and the power distribution units (PDUs) for measuring the power consumption of physical nodes and showing the connection of those three via Zabbix to the three separated layers. Zabbix is used as monitoring system in order to measure parameters defined in the monitoring metrics. To perform all necessary calculations being defined in the set of monitoring metrics the Zabbix server (infrastructure) and the aggregator (Zabbix at VM level) need to be able to communicate to each other. More precisely, Zabbix server collects the monitoring information for the infrastructure, whereby a Zabbix server is installed at each infrastructure site. Further, Zabbix aggregators are used to collect energy metrics for the virtual machines (VMs). For each application being executed in the cloud a Zabbix aggregator is created. However, the monitoring metrics will be explained in the next chapters.

Based on the measured values the monitoring metrics define not only the parameters but as well the performed calculations. Thus, calculations are necessary on the infrastructure layer, distinguishing between infrastructure sites and physical nodes, the virtual layer, meaning the virtual machines hosted on the physical nodes, and the application layer, being the services hosted in the VMs. The ECO$_2$Clouds monitoring infrastructure enables the monitoring and calculation of the required parameters as described in the monitoring metrics. It is presented in Figure 2 (see D3.3).

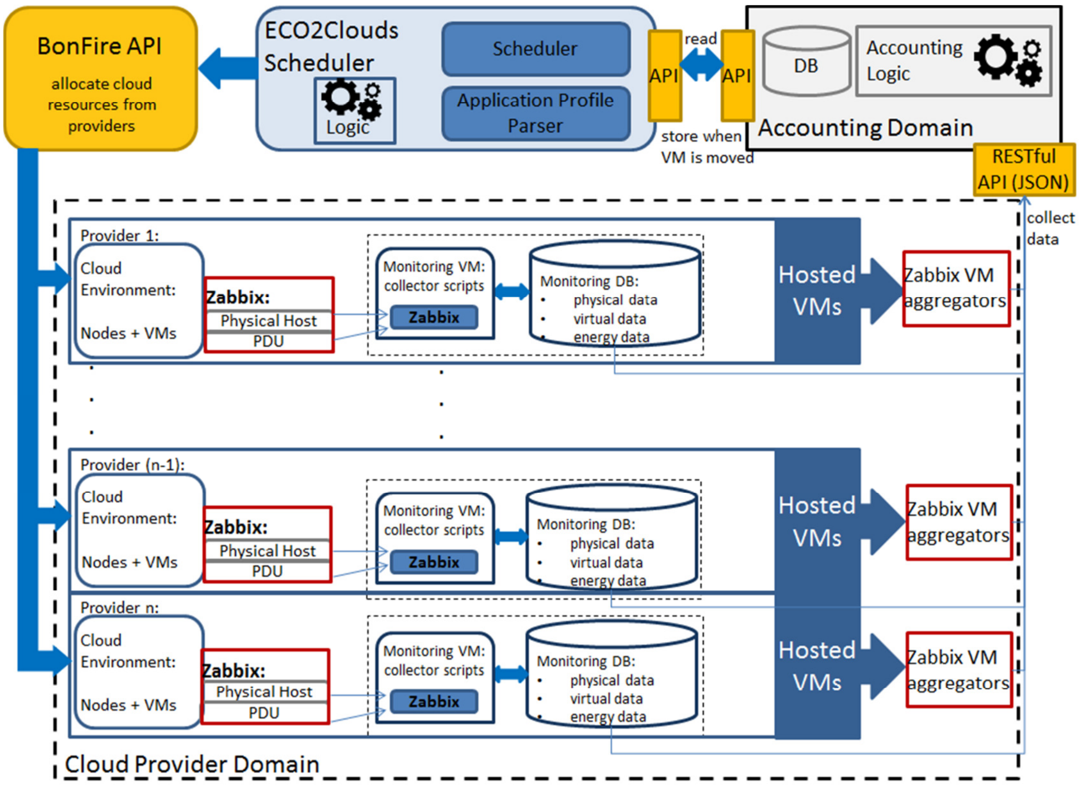


Figure 2: ECO$_2$Clouds Monitoring Infrastructure

To give a brief overview, each provider has installed a Zabbix server on a physical node in its infrastructure. The Zabbix server monitors the physical nodes and the power consumption by using the PDUs. The Zabbix server is installed in a dedicated monitoring VM and stores the monitored data in the monitoring database (DB). The VMs are hosted on the infrastructure being monitored by the aggregators (Zabbix). The collected data are stored in the Accounting DB via a RESTful JSON API. Further, the ECO$_2$Clouds scheduler is linked to the accounting domain and thus being enabled to receive the monitoring data. In addition to the above presented monitoring architecture, Section 4 presents the detailed monitoring data flow of components being part of the ECO$_2$Clouds project architecture.

The detailed description of the monitoring architecture, its components, structure and usage is given in D3.3. In addition, D3.3 presented the first sketch of the Data Mining approach. The detailed Data Mining approach is presented in Section 7. Since the finalization of D3.3 we focused on the following tasks:

- Defining the data flow between components relevant for the monitoring
    - Identifying relevant components

- Updating component implementations

- Finalization of the used metric set
    - updating the energy mix metrics and used measuring units
    - concluding the final set of used metrics
    - shift of metrics to other layers or components if metric calculations were trivial and not related to parameters from other metric calculations
- Refinement of the implemented metrics
    - General update of the implementations
    - Aligning the implementation of the power consumption / VM metric with the latest improvements taking into account the calculation of the carbon footprint estimation of providers

Furthermore, the data mining approach was moved from a sketch to the final data mining architecture. This included the following tasks:

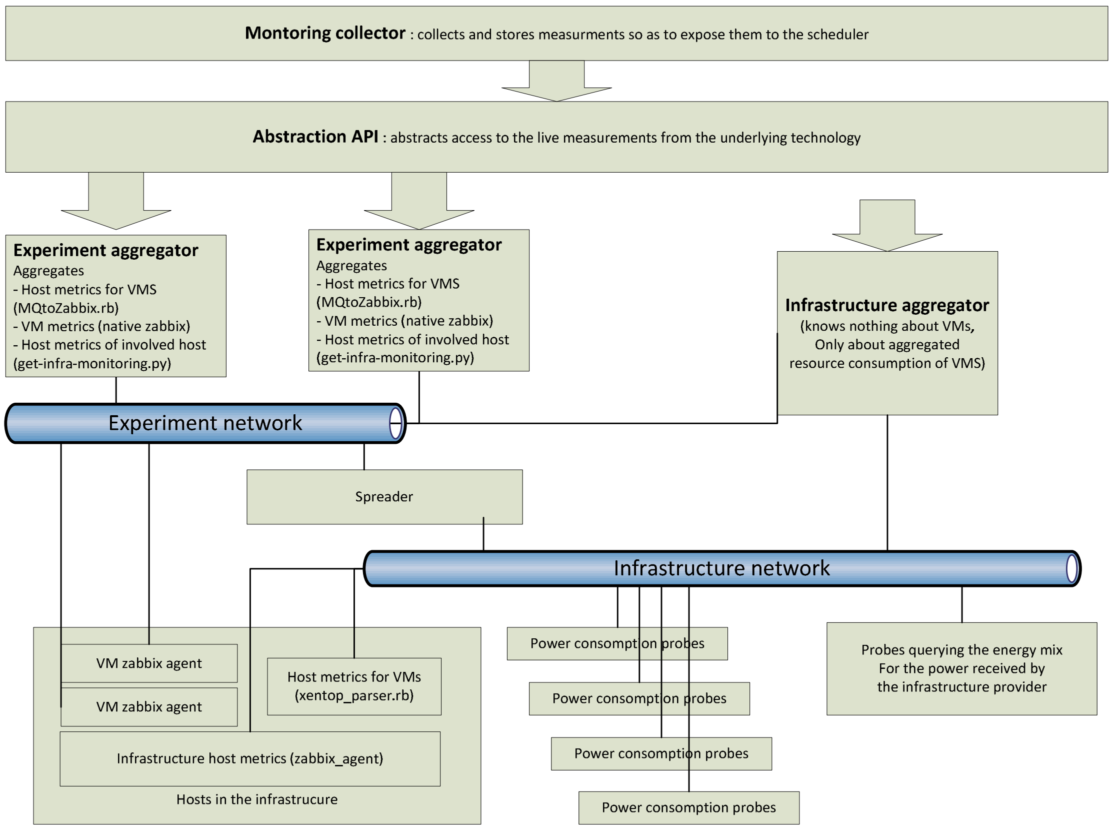
- Finalization of the data mining architecture
- Developing the concept and implementing the data mining approach
- Development of the data aggregation and reduction approach

4 Monitoring Components

Besides the monitoring infrastructure, presented briefly in the last section, the monitoring approach is presented in the documents D3.2 and D3.3 and additionally D3.3 is explaining how to use the monitoring infrastructure.

Further, on the one hand the monitoring infrastructure is the foundation for the monitoring approach and on the other hand we need to identify and define the monitoring metrics data flow across the borders of WP3. In the following, components involved in the monitoring data flow are presented (*Zabbix*, *BonFIRE Abstraction API*, *Monitoring Collector* and *Accounting Service API*).

Figure 3 presents the monitoring metrics data flow between involved components.



Montoring collector : collects and stores measurments so as to expose them to the scheduler
Abstraction API : abstracts access to the live measurements from the underlying technology
Experiment aggregator
Aggregates
- Host metrics for VMS (MQtoZabbix.rb)
- VM metrics (native zabbix)
- Host metrics of involved host (get-infra-monitoring.py)
Experiment aggregator
Aggregates
- Host metrics for VMS (MQtoZabbix.rb)
- VM metrics (native zabbix)
- Host metrics of involved host (get-infra-monitoring.py)
Infrastructure aggregator
(knows nothing about VMs, Only about aggregated resource consumption of VMS)
Experiment network
Spreader
Infrastructure network
VM zabbix agent
VM zabbix agent
Host metrics for VMs (xentop_parser.rb)
Infrastructure host metrics (zabbix_agent)
Hosts in the infrastrucure
Power consomption probes
Power consomption probes
Power consomption probes
Power consomption probes
Probes querying the energy mix For the power received by the infrastructure provider

Figure 3: Monitoring Components

The overall system architecture includes an experiment and an infrastructure network. The infrastructure network gathers information related to the energy mix of data centres, the power consumption of the infrastructure and the hosts in the infrastructure. Further, the hosts in the infrastructure are connected to the experiment network and in addition the infrastructure network is connected to the experiment network via a spreader component.

The BonFIRE Abstraction API accesses the experiment and the infrastructure aggregators in order to receive the live measurements of the underlying monitoring. The Monitoring Collector accesses the BonFIRE Abstraction API for collecting and storing the measurements and making them available in the accounting domain.

The details regarding the components relevant for the monitoring are presented in the next sections.

4.1 Zabbix

The monitoring infrastructure uses Zabbix; a Zabbix server on the infrastructure layer (infrastructure aggregator) and a Zabbix aggregator for each cloud application (virtualization layer) running in the federated infrastructure (experiment aggregator). This approach is briefly summarized in the previous section and in the documents D3.2 and D3.3 and additionally D3.3 is explaining how to use the monitoring infrastructure.

For all Zabbix server and aggregator running in the federated cloud infrastructure, the monitored data is stored in the Zabbix DB. As presented in D3.3 the Zabbix DB is a MySQL DB. The data stored in the Zabbix DB are required by the BonFIRE Abstraction API.

4.2 BonFIRE Abstraction API

The BonFIRE abstraction API's role is to publish lists of metrics and to give access to them (latest value and history) independently from the collection infrastructure used (Zabbix in BonFIRE's case). It therefore serves as a generic layer sitting between the infrastructures used by the ECO$_2$Clouds Scheduler and the testbeds it uses. This abstraction API can be implemented by other testbeds willing to adopt the ECO$_2$Clouds solution.

The API publishes a list of infrastructure metrics, a list of metrics available on each host of the infrastructure and a list of supported metrics for all VMs running on the infrastructure. For each of these metrics, it is possible to retrieve the latest value as stored in the infrastructure aggregators or experiment aggregators, and to get access to and range of values in the past.

It does not store any information. When the experiment aggregators are discarded, the abstraction API cannot give access to values they had aggregated. This is the role of the Monitoring Collector.

4.3 Monitoring Collector

The ECO$_2$Clouds solution requires the collection of numerous metrics from cloud infrastructures. The metrics are used by real time components (like the ECO$_2$Clouds 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. Furthermore, ECO$_2$Clouds should be able to collect metrics from various cloud infrastructures.

The ECO$_2$Clouds 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 ECO$_2$Clouds Metrics Abstraction REST API (Section 4.2). 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.

4.3.1 Workflow of the ECO$_2$Clouds Monitoring Collector

As we mentioned earlier, the Monitoring Collector is responsible for collecting metrics from the cloud infrastructure and cloud status updates. The Monitoring Collector utilises the ECO2Clouds Metrics Abstraction REST API (see Section 4.2) 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.

The Accounting DB is the persistent storage medium of ECO$_2$Clouds. The Accounting DB is able to reply to requests from the real time components of the ECO$_2$Clouds solution (like the scheduler) and from the data mining and aggregation components. The Accounting DB was designed to handle large amounts of data and to allow future expansion. The Accounting DB allows new metrics to be added without any impact to the schema of the database or to the source code of the Monitoring Collector. Furthermore, the Accounting DB is used as a logging mechanism for the ECO$_2$Clouds Scheduler. Every action of the scheduler is stored in the database and can be reviewed for correctness.

4.3.2 Implementation of the ECO$_2$Clouds Monitoring Collector

The implementation of the ECO$_2$Clouds Monitoring Collector is depicted on Figure 4.

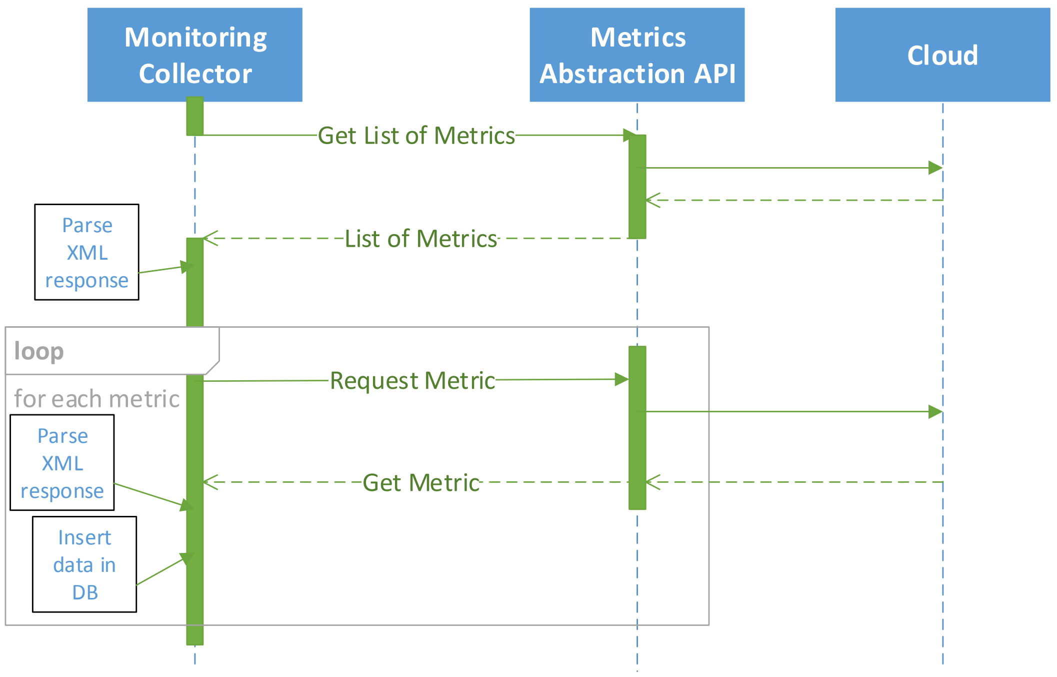


Figure 4: Monitoring Collector Sequence Diagram

The Monitoring Collector runs a never-ending loop, requesting metrics from the cloud infrastructure for the three abstraction levels: Site, Physical Host and VM. The first call requests a list of resources of a specific type (sites, physical hosts, VMs). The reply contains the list of resources. Then, for each resource in the list the Monitoring Collector requests a list of available metrics. The response from the cloud provider contains a list of metrics. Finally, a request is send for each metric. Each reply is parsed and stored in the Accounting DB. Due to the instability of network communications, the Monitoring Collector is using a very defensive style of programming. Each reply from the cloud provider is checked for completeness and consistency. If it fails any checks, then that metric is ignored and not stored in the Accounting DB.

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 ECO$_2$Clouds Scheduler or other ECO$_2$Clouds components query the Accounting DB in order to retrieve and process the event.

The initial design stated that the Monitoring Collector is a simple double threaded stateless REST API client. One thread was used to gather metrics and the other thread to gather status updates. However, during the course of the project, we realised that the metrics thread had a frequency of 25 minutes for a 3 site, 47 Physical Host and a couple of VMs installation. A 25 minute sampling rate was not enough for the Scheduler and other real time ECO$_2$Clouds components. The main bottleneck was the serialisation of the metrics gathering process. We spend considerable effort in re-designing the Monitoring Collector into a multithreaded application. Once the re-design was complete, the Monitoring Collector was using a separate thread for each physical host and for each VM. Each physical host and VM thread spawns separate threads for each metric. Thus, any network delays or hiccups in the cloud are hidden. Once we were satisfied with the performance of the multithreaded design in offline unit testing, we run the Monitoring Collector against a real cloud infrastructure. We expected a huge reduction of the sampling rate. The sampling rate was reduced to 8 minutes; however, the cloud infrastructure was not able to cope with the amount of requests. As a result, we refactored the design and the Monitoring Collector is now using a pool of threads. The pool of threads does not allow more than 10 concurrent threads to be active. Using 10 threads, the sampling rate is 5 minutes which provides enough accuracy for the real time components of the ECO$_2$Clouds solution and stability for the cloud infrastructure.

The initial design of the Monitoring Collector did not take into account the possibility that the Monitoring Collector would need to restart (either due to an error or to an update). Thus, the Monitoring Collector did not query the Accounting DB or the cloud infrastructure for the current state of VMs. Instead it started with a clean slate and did not monitor any VMs. This behaviour caused a lot of issues with ECO$_2$Clouds testers, since they had to re-initialise their VMs every time the Monitoring Collector restarted. The code was refactored and the Monitoring Collector is now attempting a two-step validation and synchronisation of the Accounting DB with the cloud infrastructure. Furthermore, an occasional restart of the Monitoring Collector does not lead to a considerable loss of metrics or events. Only the metrics and events during the duration of the restart are lost.

4.4 Accounting Service

The ECO2Clouds component responsible to expose the monitoring information to the ECO2Clouds Scheduler it is the ECO2Clouds Accounting Service. This component exposes a REST interface from which the ECO2Clouds Scheduler can query to get different energy metrics from the VMs running in the infrastructure.

4.4.1 Architecture and Workflow

Checking the deliverable D2.4 (D2.4 2014), the architecture block diagram for the ECO2Clouds Accounting Service presented the following shape:

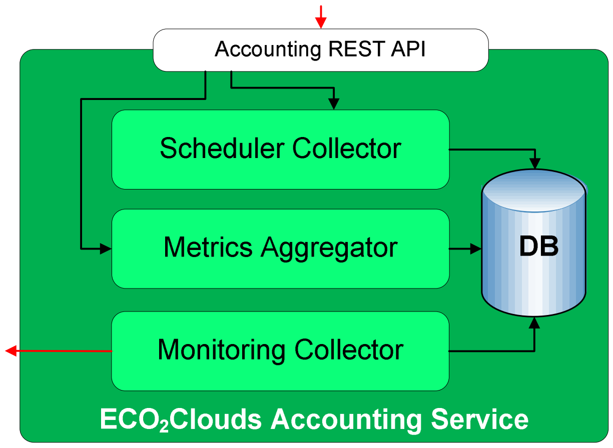


Figure 5: Accounting Service Architecture

Metrics Aggregator component, shown in the previous figure, will connect with the same database where the Monitoring Collector stores the different metrics coming from the different federated cloud facilities. The mission of this component it is perform the different queries coming from the Accounting REST API. Those queries can be performed by four different categories:

- /locations/<location-name>/monitoring – A query to this path in the REST interface will retrieve the latest monitoring values for a specific Cloud site (specify by "location-name" variable).
- /locations/<location-name>/hosts/<physical-host>/monitoring – A query to this path in the REST interface will retrieve the latest monitoring values for an specific physical host (defined by the "physical-host" variable), ie. Infrastructure layer, in a specific Cloud site (defined by the "location-name" variable.
- /experiments/<experiment-id>/locations/<location-name>/vms/<vm-id> - A query to this path in the REST interface will retrieve the latest monitoring values for an specific Virtual Machine (defined by the "vm-id" variable), in an specific Cloud site (defined by the "location-name" variable), associated to an specific experiment/application execution in ECO2Clouds Scheduler (defined by the "experiment-id" variable").
- /experiments/<experiment-id>/monitoring – A query to this path in the REST interface will retrieve the specific application monitoring

information.

All the previous queries could be modified with two query params: startDate and endDate to get a collection of monitoring values instead of only the last ones.

Once the Scheduler inputs a query to the Accounting Service REST interface, it is the mission of the Metrics Aggregator to perform the different queries to the Monitoring Collector DB to retrieve the specified monitoring information, aggregates them, and sends the results back to the user.

The only component that can interact with the Accounting Service REST API it is the ECO$_2$Clouds Scheduler. The Scheduler acts as proxy for other ECO$_2$Clouds component, such as Portal and Application Controller, or the user. While the Accounting Service REST API trust any request coming from the Scheduler, the Scheduler does not trust any request coming to its REST interface. ECO$_2$Clouds Portal and ECO$_2$Clouds Application Controller need to send a valid user and password, also, they should have the rights to access to that specific information (only in the case of application or VM metrics).

The REST path that the other components or the user need to specify to the Scheduler to access to the monitoring information matches the path format shown before for the Accounting Service.

4.4.2 Derivative metrics

Section 5.3 presents a series of metrics definition that are derivative from other metrics collected by the Monitoring Collector. It is the responsibility of the Monitoring Aggregator to calculate those metrics, if necessary, each time the Scheduler makes a request for them by the Accounting Service REST API.

5 Metrics

In the D3.2 (D3.2 2013) document the monitoring infrastructure and metrics were presented and a general refinement of the realization of the monitoring approach and implementation was presented in D3.3. As stated out briefly in the last section, the monitoring metrics are required for the monitoring in order to measure relevant parameters and perform calculations based on them. Relevant parameters are those being specific indicators, e.g. for performance, workload or power consumption, of the physical nodes and the virtual machines. The calculations are necessary for making use of the measured parameters and making assumptions regarding the behaviour of the physical nodes and the virtual machines.

This section covers the monitoring metric refinements during the final development phase.

5.1 Refinements

The set of monitoring metrics reflects the energy efficiency of IT systems from a holistic perspective opposed to a pure infrastructure based monitoring system. The ECO$_2$Clouds monitoring approach covers the mentioned three layers, infrastructure, virtualization and application. In addition, this approach allows the derivation of the interrelation between different components of IT cloud infrastructure. The ECO$_2$Clouds approach is usable at different provider sites with different hardware configurations and different energy sources.

The ECO$_2$Clouds project follows a layered approach: the application layer, the virtual machine (VM) layer and the infrastructure layer. Further, the following section presents the final set of metrics for each layer.

5.1.1 The Application Layer

The application layer metrics are customized for the specific needs of user applications.

Table 1 presents the name, definition and units for the selected metrics for the application layer.

Table 1: Application layer metrics

Metric	Definition	Units
Task execution time	The time taken to execute the specific task.	s
Application execution time	The time taken to execute the whole application.	s
Power consumption	The power currently consumed by the analysed application. For simplicity, this metric is derived by aggregating the power consumption of the VMs hosting / operating the application.	W

Response Time	The average time taken to handle user requests. In particular, it measures the average time interval from a user request to the service response. This metric is particularly relevant for interactive applications; for batch applications the response time will coincide with the application execution time.	s
Throughput	Rate of executions of an application. For example, for the ECO$_2$Clouds application this metric is calculated as the ration between the amount of operations executed in a specific time interval (e.g., 30 seconds).	s^{-1}
A-PUE (Application PUE)	The application power usage effectiveness (PUE) is the ratio between the total amount of power (*P)* required by all VMs of an application *i* and the power used to execute the j-th application task: $$A - PUE_i = \frac{\sum_k P_{ik}}{\sum_{jk} P_{ijk}}$$	None
Application Energy Productivity (A-EP)	The Application Energy Productivity is the ratio between the number of executions of the tasks hosted by all VMs of application *i* and the total energy for the execution of the VM: $$A - EP_i = \frac{NTrans_{i\Delta t}}{\sum_k (P_{ijk} * \Delta t)}$$	W^{-1}
Application Green Efficiency **(A-GE)**	The Application Green Efficiency metric measures how much green energy is used to run application *i*. We multiply the power consumed by all VMs of application *i* by the percentage of used green energy. $$A - GE_i = \sum_k GEC * (P_{ik} * \Delta t)$$	W

5.1.2 The Virtual Machine Layer

Table 2 presents the selected metrics for the VM layer, their definition and the used units.

Table 2: VM layer metrics

Metric	**Definition**	**Unit**
CPU usage	Current processor utilization percentage measured / as seen by the running virtual machine. It is measured as the ratio between the amount of used CPU over the amount of allocated CPU.	%

Storage usage	Storage utilization (S_{usage}) on the corresponding virtual storage device is defined as the ratio of the used disk space (S_{used}) to the total amount of disk space (S_{total}). $$S_{usage} = \frac{S_{used}}{S_{total}} * 100$$	%
I/O usage	The percentage of the process execution time in which the disk is busy with read/write activity. The calculation sums the read and write activities in a given time interval and divides it by the duration of the time interval.	%
Memory usage	Ratio of the used memory of the virtual machine and the available amount of memory. The calculation divides the used memory by the available memory.	%
Power consumption	The power consumed by the analysed VM at a given time. The ECO2Clouds virtual machine power consumption formula defines the consumed power per virtual machine P_{VM} as the sum of the VM contribution to keep the host running plus the actual VM consumption, as follows: $$P_{VM} = \frac{P_{HostIdle}}{\#VM} + (P_{Host} - P_{HostIdle}) * CPU\%_{VM}$$ The first term is calculated by dividing: the idle power of the physical host ($P_{HostIdle}$), as measured on the site PDUs; by the number of running VMs ($\#VM$). For the actual VM consumption term, the net physical host power consumption ($P_{Host} - P_{HostIdle}$) is calculated and multiplied by the CPU utilization of the VM as seen from the physical host ($CPU\%_{VM}$). In turn, the CPU utilization of the VM as seen by the physical host is calculated by dividing: the amount of CPU dedicated to the VM in the last measurement interval (ΔCPU_{VM}); by the sum of the CPU dedicated to each VM in the last measurement interval. $$CPU\%_{VM} = \frac{\Delta CPU_{VM}}{\sum_i \Delta CPU_{VM_i}}$$	W
Disk IOPS	This metric assesses the disk input / output rate of a virtual resource at a given time. The calculation sums the read- and write-rates reported by the operating system at the specified time, for all the disks mounted on the VM.	OPS/s

5.1.3 The Infrastructure Layer

The infrastructure metrics are available for the infrastructure site and the hosts. This separation was performed because of the different nature of both. A site metric is used for a complete provider site while a host metric is dedicated to a specific physical host, e.g. the Site utilization metric takes into account the total and the available amount of cores of a site while the cores are associated to a specific host. The host metrics are specified to the physical nodes, e.g. the power consumption metric is a host metrics because we need to know the consumed power for each physical node.

In addition to the infrastructure site and host metrics, each site measures the energy mix metrics as well. Table 3 describes the metrics about the energy mix, the site metrics are presented in Table 4 and the host metrics are presented in Table 5.

Table 3: Infrastructure layer energy metrics

Metric	Definition	Unit
Energy Mix	This metric measures the amount of different energy mix components for the power grid provider. The following components are considered: Biomass, CCGT (Combined Cycle Gas Turbine), Coal, Cogeneration (of heat and power), Fossil, Gas, Geothermal, Hydraulic, NPS hydro, Nuclear, OCGT (Open Cycle Gas Turbine), Oil, Other, Pumped storage, Solar, Total green, Water and Wind	%
Produced CO_2	This metric calculates the amount of CO_2 which is necessary to produce 1 kWh. Due to contractual issues, HLRS is providing a fixed value for this metric whereas EPCC and INRIA are using live data from their power grid provider.	g/kWh
Grid total	Grid total (optional metric used at EPCC and INRIA) measures the total amount of produced power by the power provider / power grid operator.	MW
Imported / Exported	Imported and Exported power by the power grid provider.	%

Table 4: Infrastructure layer metrics-site

Metric	Definition	Unit

Site utilization	Current utilisation of a single site. The metric is defined as the ratio of available free cores of all worker nodes; over the total number of cores of all worker nodes. In ECO$_2$Clouds, this metric is gathered on the frontend (OpenNebula cloud manager) of a site.	%
Storage utilization	The percentage of the frontend storage usage gives a good representation of the storage site usage (as the frontend hosts all permanently stored images). If necessary, this metric can be implemented for single worker hosts as well.	%
Availability	A site will be available if the Cloud API server provides a reasonable and well-defined answer to a request. Furthermore, at least one worker host has to be available (see definition on Table 5). In ECO$_2$Clouds this metric is composed by three different sub-metrics: 1) The number of hosts seen as available by OpenNebula (return number or 0). 2) Check the TCP port 8443 where the OCCI server of the frontend is reachable (return 1 or 0). 3) Check the TCP port 4567 where the OCCI server of the frontend is listening (return 1 or 0) The three sub-metrics are combined, so if one is zero, the site is unavailable.	Boolean
PUE	The Power Usage Effectiveness metric is used to determine the energy efficiency of a site. It is measured as the ratio of the power that is used by the computing equipment over the total facility power In ECO$_2$Clouds practice, this metric is represented by a static value, different at each site.	None

Table 5: Infrastructure layer metrics-host

Metric	Definition	Unit
Power consumption	The power consumed by a specific host at a given time.	W

Disk IOPS	This metric checks the rate of input / output operations of the disk within a physical host. This metric includes the operations of all virtual instances running on the host and those of the underlying operating system.	OPS/s
CPU utilization	The utilization of the processor(s) inside a physical host. For each host this metric indicates how much processor capacity is used for the underlying operating system and all the virtual instances at a given time.	%
Availability	A host is available if it is seen as "online" within the cloud manager at the site frontend.	Boolean

5.2 Common Practices and Problems

ECO2Clouds developed tools to allow all participating BonFIRE sites to easily expose the required metrics. However, the differences of hardware configurations and energy mix policies concluded in issues that had to be overcome by a project strategy fitting to all ECO2Clouds partners. These are discussed below.

5.2.1 Common practices

In order for the implementation of the monitoring metrics to be adoptable at all provider sites, Zabbix templates, bash, python and Ruby scripts were implemented and shared between the project partners. The monitoring metrics are aligned to all providers' requirements taking into account different hardware configurations and different conditions regarding the used energy mix of the providers.

5.2.2 Implementation challenges

1. Different hardware

The hardware nodes available to the ECO2Clouds consortium are provided by EPCC, HLRS and INRIA. Thus, different hardware with different configurations is used even within each providers domain, e.g. at HLRS dual core, quad core, 12 core in also single socket, dual socket and quad socket with different configurations are used. The used nodes at EPCC, HLRS and INRIA are listed in the BonFIRE infrastructure documentation (BonFIRE 2014).

The different hardware concludes in hardware nodes responding very fast to received requests and others responding very slowly. In the past, especially for the second case it could happen that Zabbix requests could not be handled because of time outs due to hardware already blocked by other requests and not able to respond. Thus, during the development phase of the monitoring metrics, ECO2Clouds partners agreed on reducing the work to be done on infrastructure level to the pure monitoring of customized monitoring parameters. Hence, simple calculations such as operations like *1 / x* were shifted to other layers, e.g. the accounting layer, whereas metrics that involve various parameters stay on the Zabbix host.

To be more precise, three metrics (*Green Efficiency Coefficient (GEC), Site Infrastructure Efficiency and Carbon Usage Effectiveness (CUE)*) are relocated to the Accounting service because they are simple calculations. By doing these calculations in the Accounting domain, the general amount of data to be stored in the monitoring DB is reduced and additionally calculation operations don't need to be performed on infrastructure level.

2. Energy mix

   a) Different energy mix rules

   The list of used metrics includes metrics for monitoring the energy mix at each provider's site. However, due to contracts with energy providers at EPCC and INRIA the energy mix is dynamic while at HLRS it is static. Through this, the energy mix metrics needed to be adapted to the needs of every provider site. In detail, for EPCC and INRIA those metrics include measurement options for receiving dynamically the current energy mix values while the same set of metrics includes static values for HLRS.

   b) Power consumption VM

   The power consumption metric for VMs was initially constructed as a general metric for exposing the used power per VM. However, in order to implement the described power consumption VM metric it was necessary to take care of the different hardware constraints of each hardware provider because of the need to consider the power consumption of physical nodes for the power consumption per VM metric. This means that the power consumption per VM calculation needs to include parameters from two layers, the infrastructure and the virtualization layer. This procedure requires updates of the hardware configuration (e.g. updating Zabbix configuration files) and the usage of a Ruby script connecting to the corresponding hardware configuration. Generally, the power consumption per VM metric was presented initially in D3.2 and concludes in D3.4 in the following sections. Section 6 describes the metrics for the carbon footprint estimation used by each infrastructure provider. It includes details regarding the implementation of the power consumption per VM metric.

To sum up, the implemented monitoring metrics are the ones described in this document. They are applicable to all infrastructures of the partners.

5.3 Metrics Moved to Accounting

With regard to Section 5.1 some of the metrics defined in D3.3 were relocated to the ECO$_2$Clouds Accounting service due to issues making them inefficient to expose directly from the monitoring infrastructure. Some other the metrics were dropped because they are redundant.

These metrics are listed and discussed in Table 6.

Table 6: Skipped / dropped metrics

Metric	Skipped / Dropped
Green Efficiency Coefficient (GEC)	Percentage of energy consumed by the site that is produced by green energy sources. This metric is calculated as the ratio between the green energy consumed by the site and the total energy consumed by the site. For the assessment of this metric, in the ECO$_2$Clouds project, information about the mix of energy sources used in the national grids are considered. This metric won't be regarded any more on the monitoring layer. It will be calculated by the accounting service presented in the monitoring architecture.
Site Infrastructure Efficiency (SIE)	This metric is used to calculate the energy efficiency of a site. SIE is the percentage value derived by dividing information technology equipment power by total facility power. This metric won't be regarded any more on the monitoring layer. It will be calculated by the accounting service.
Carbon Usage Effectiveness (CUE)	The CUE formula is CEF * PUE where CEF is the Carbon Dioxide Emission Factor (gCO2EKWh). CEF is calculated analyzing the site energy mix. In fact, it is the weighted average of the CEFs related to the energy sources used in the site. CEFs of the sources are taken from the literature. This metric won't be regarded any more on the monitoring layer. It will be calculated by the accounting service.
Site Energy Productivity	The productivity of a site is measured as the ratio between the work output (in the project the work output is the number of executions of the tasks) and the energy consumed by the site. This metric is dropped on infrastructure site level because no further calculations are required for this.
Site saturation	The degree in which the site computing resources are used. This can be measured by considering the aggregation of the host utilization indexes. This metric is dropped on infrastructure site level because no further calculations are required for this.

Green Energy saturation	The degree to which green energy sources are used with respect to the capacity of the site. This metric is only assessable for Germany due to the fixed amount of green power available. This metric is dropped on infrastructure site level because the GEC allows required calculations in the accounting domain.

6 Carbon Footprint Estimation by Providers

6.1 Overview

As an extension to the ideas developed in ECO$_2$Clouds, the project has decided to study shifting carbon footprint estimation from the ECO$_2$Clouds scheduler to the infrastructure providers, so as to allow building an eco-system where

- Infrastructure providers compete to expose the lowest unit cost (in the sense of carbon footprint costs) to attract users. They can compete by reducing the carbon footprint of their datacenter or on the strategy used to ensure that the billed cost and the real cost of using their infrastructure stay aligned. One provider could over-estimate by default so as to ensure billed cost is always over real costs, and use the difference to attract new customers or under-estimate costs for premium customers. Another provider could optimize announced CO_2 cost by taking into account the weather forecast, usage patterns at the same time the previous week or previous year or forecasted CO_2 impact of energy estimated by their electricity provider.
- Schedulers or other cloud brokering services base their provisioning decisions on costs known in advance, and revise their strategy when the cost is updated. They can therefore compete on the services provided to applications to characterise their usage and in adaptation actions implemented on behalf of user-specified strategies, so as to keep the cost minimal.
- Certification agencies can compare billed CO_2 cost to users to the real measured cost, so as to certify that all computing done on a given provider's resources are pushed back to customers.

In the timeframe of the ECO$_2$Clouds project, we will demonstrate this approach by having all infrastructure providers publish real and estimated CO_2 unit costs for CPU, memory, disk and network resource usage.

This implies support for the following additional metrics at host level:

Table 7: Additional host level metrics

Metric	Definition	Unit
Total amount of cpu.seconds used by running VMs	The amount of CPU-seconds consumed by the running VMs	s
Total memory usage by VMs	The total memory usage of running VMs	Byte
Total incoming network traffic from all VMs	The total network activity caused by running VMs. This metric includes the network receive activities.	Byte/s
Total outgoing network traffic from all VMs	The total network activity caused by running VMs. This metric includes the network send activities.	Byte/s
Total amount of disk reads from VMs	The total disk activity caused by the running VMs. This metric includes only disk read operations.	None
Total amount of disk writes from VMs	The total disk activity caused by the running VMs. This metric includes only disk write operations.	None

6.2 Implementation

It is possible to collect, for all physical hosts, metrics totalling resource consumption by user VMs. Specifically, for each host, infrastructure providers measure the following metrics:

- Number of running VMs
- Total CPU-seconds consumed by running VMs
- Total memory usage of running VMs
- Total network activity (sent and received) due to running VMs
- Total disk activity (write and read) due to running VMs

To estimate the unit cost of each operation, providers will aggregate these values for all hosts in their testbed. Therefore, for each testbed, the total amount of CPU-seconds, memory usage, network usage and disk usage attributable to VMs will be collected across all hosts. At the same time, power consumption, and CO_2 impact of power consumption is collected. The infrastructures generally include heterogeneous hardware. In this discussion we group sets of identical hosts in one site as clusters and perform this data aggregation for each cluster of identical hosts.

Providers will then attribute the following:

- Power consumption of infrastructure (hosts, storage equipment and network equipment) to each cluster. This can either be done dynamically using the ratio of power usage for each cluster, or statically to account for external factors (such as better different stress impact on the infrastructure of each cluster because of different network connectivity or reliance on storage external to the cluster. For example if a provider has a cluster of 10 hosts and a cluster of 15 hosts, 40[1]% of infrastructure power consumption could be added to the total power consumption of the first cluster, and 60% to the second cluster.

- A percentage of power consumption of hosts to each category of resource usage. For example, 80% of power consumption can be attributed to CPU usage, 10% to memory usage, 5% to network traffic and 5% to disk IO. It is up to the providers to do their own research so as to make these attributions in line with the characteristics of their hardware or business model.

Using this information, providers can dynamically calculate an average unit cost for resource consumption.

They will publish such a unit cost, and update it regularly (every 6 hours for example). This allows application providers, who are able to estimate the resource consumption of their application expressed in CPU-seconds, memory required, network and disk I/O units, to estimate the cost in CO_2 of running their application on a provider's infrastructure.

It is suggested here that infrastructure providers then bill CO_2 usage using the published estimation, and compensate for the difference between real CO_2 impact and billed CO_2 impact by feeding the difference in the computation of the next estimation. A provider will only be certified (as trustworthy for the ECO2Clouds scheduler for example) if over a monthly period, the difference between real CO_2 impact (computed dynamically) and billed CO_2 impact stays under a given threshold.

[1] 10 / (10+15) = 0.4.

7 Data Mining Service

In this section we describe the ECO₂Clouds Data Mining service (DM Service). The motivation, goals and preliminary concepts of the DM Service were already presented in D3.2 [D3.2 2013] and D3.3 [D3.3 2013]. Here, we present some modifications of its architecture and concepts, and describe the implementation and algorithms in more detail.

7.1 Architecture

The ECO₂Clouds Data Mining service consists of two major components (see Figure 6). The first component is running on the Accounting Service to perform transfer of non-reduced metrics data to a remote data storage (DM Storage), and to generate a reduced data set. By this we avoid uncontrollable growth of the Accounting DB, which could cause overloading with the old metrics data while the new data could not be stored and used by Scheduler.

The second component is DM Storage, which gathers the non-reduced metrics data and performs statistical analysis over them, e.g., correlation analysis over a large enough portion of data. The resulting stable parameters of this analysis are inserted into the Accounting DB as a separate table.

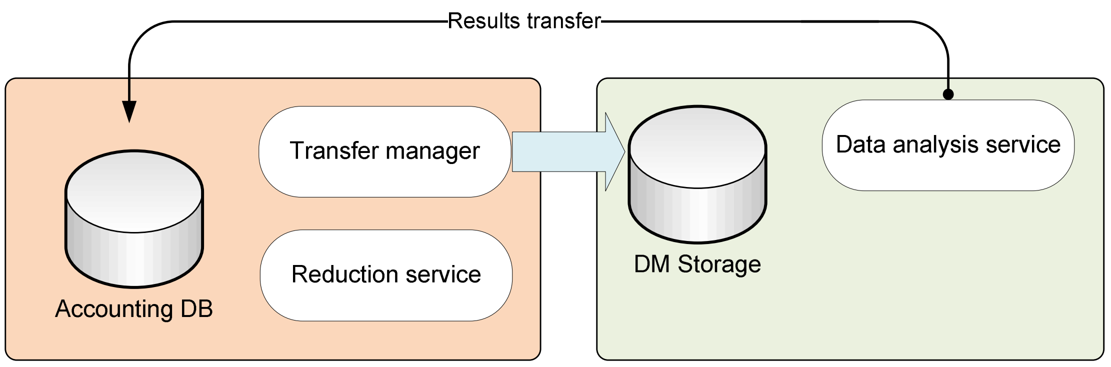


Figure 6: Data Mining Architecture

7.2 Concept

Here we present the Data Mining workflow (see Figure 7), since it includes some changes compared to the version from the previous deliverable D3.2 [D3.2 2013] and D3.3 [D3.3 2013]. The major update is that the reduction operation is shifted to the Accounting host because of the hardware restrictions on the DM Storage side. The updated workflow is as follows:

1) Triggered daily, the Accounting Service inserts the new non-reduced data into the database of DM Storage;
2) Triggered daily, the Accounting Service performs the reduction of the metrics data; the results of data reduction are inserted into a separate database on the Accounting VM;
3) The initial non-reduced data are deleted – thus, each time the Accounting Service has to perform operations over a metrics data set which was produced only during the <u>last 24 hours</u>; the loss of data does not affect the Scheduler functionality, because if does not use the historical data for

physical hosts, while for experiments we only reduce the data of finished (non-active) experiments only.

4) On the DM Storage side, the statistical analysis of the metrics data is performed;

5) The results of the statistical analysis are inserted into Accounting DB as a separate table.

7.3 Implementation

On the Accounting VM, we install a bash script which performs the following operations:

1) Creates a temporary SQL-dump of the current non-reduced “e2c_collector” database;

2) Imports the generated SQL-dump into DM Storage by inserting the rows incrementally to the previously stored there data;

3) Removes the temporary SQL-dump from the Accounting VM;

4) Calls the reduction service implemented as a Java program, which performs the data reduction and inserts its results into a separate database on the same VM. As presented in Figure 7, copied non-reduced data will be deleted. We will describe the data reduction process in more detail later in Section 7.4.

This script is set to be executed as a cronjob daily, at 23:59. The amount of metrics data produced daily is about 10-15 Mb. The reduction process over such data set takes around 5 minutes for reduction rate = 2 and can be much faster for higher reduction rates. Import of this information from Accounting DB to DM Storage also takes around 5 minutes.

The metrics data is being collected into “e2c_collector” schema of the Accounting DB. The fast-growing tables of “e2c_collector” include “experiments_items”, “virtual_machines_items”, “physical_hosts_items” and “sites_items”. The total number of rows added to these tables per day is more then 100 000.

On the DM Storage side, the metrics data analysis is being performed after a large enough data set is collected. More details about data analysis are presented in Section 7.5.

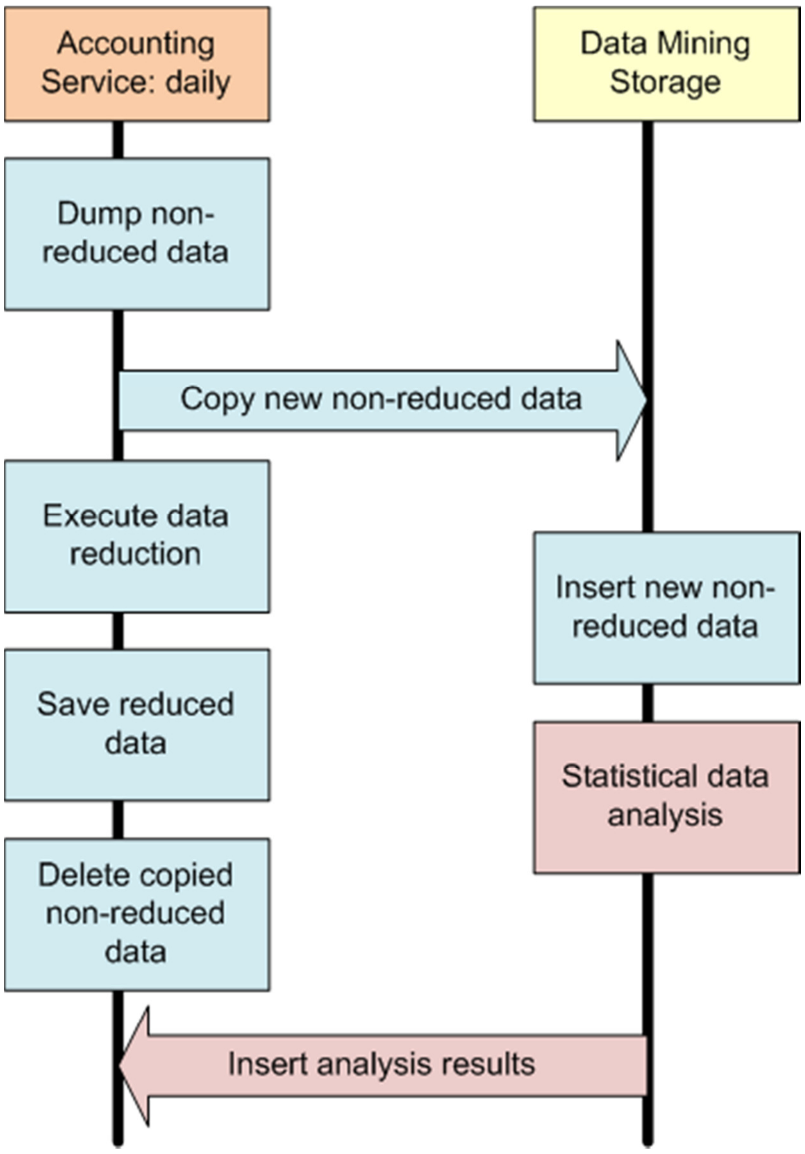

Accounting Service: daily
Data Mining Storage
Dump non-reduced data
Copy new non-reduced data
Execute data reduction
Insert new non-reduced data
Save reduced data
Statistical data analysis
Delete copied non-reduced data
Insert analysis results

Figure 7: Data Mining Process

7.4 Data Aggregation and Reduction

The reduction of the Monitoring Collector DB database is performed through aggregation conducted in the following steps (see Figure 8):

1) We are sorting the metrics items from the “items” table by (a) type (experiments, virtual machines, sites or physical hosts), (b) name of the metric, (c) location, (d) time.
2) Given the reduction rate *r*, we calculate mean value for each *r* items (with the same type and name). We make an additional check, so that the mean value is not calculated over metrics of various types, names or locations.
3) We insert the calculated mean value into the new reduced “items” table.
4) We reduce the four tables “experiments_items”, “virtual_machines_items”, “physical_hosts_items” and “sites_items” by deleting from them those entries whose IDs are not present anymore in the reduced “items” table.

Note that the remaining tables of the "e2c_collector" database remain unchanged, since they do not grow fast over time.

After reading the reduction rate from the config file as an input parameter, we adjust it depending on the current database size: if the database is smaller than 200 Mb, no reduction is performed. The reduction rate can be also defined manually in the config file depending on the user's priorities w.r.t. free space, duration of reduction, data precision, etc.

For the VM (and experiment) metrics, we are checking the status (active or not), and duration of VM (and experiment) run. If the given VM (and experiment) is not active its items are reduced according to the given reduction rate. Otherwise, the metrics items of such VM (and experiment) are not reduced.

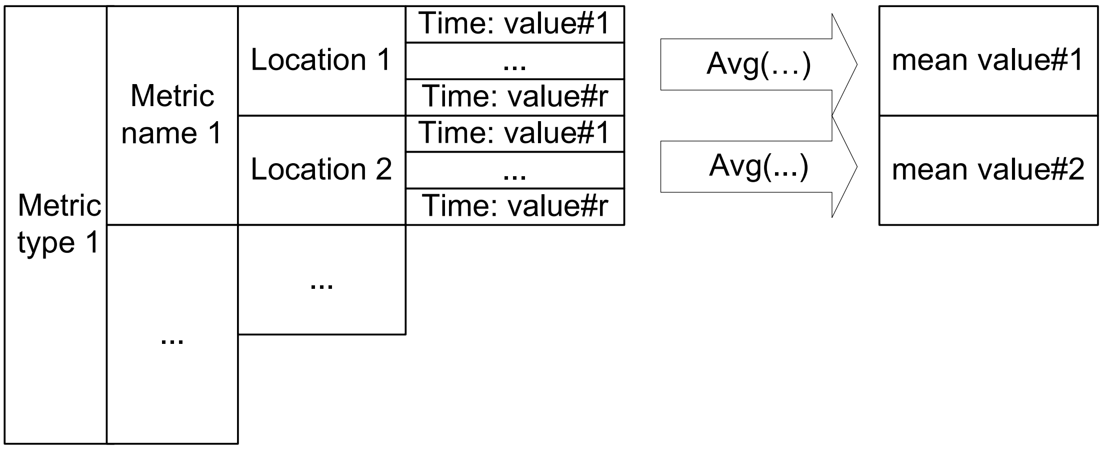


Figure 8: Data Mining Reduction

7.5 Data Analysis

The data analysis aims to identify significant patterns in data. This section introduces the types of analysis that are conducted in the ECO2Clouds project.

The inputs of the data analysis are the values of the metrics monitored within the project. The kinds of performed analysis are described in the following subsections together with their goals and initial results.

7.5.1 Correlation Analysis among Metrics

Correlations represent the strength of a linear relationship between two random variables. In our scenario, correlations are crucial in order to find relationships among metrics. Understanding the way in which the metrics are linked is useful for the components that have to select the most suitable adaptation action.

The results of correlations can be stored by using three fields: metric 1 – metric2 – correlation index. It is also possible to store only the pairs of metrics that have a strong correlation (e.g., correlation index > 0.8).

Analysing a first set of experiments, for example, on empirical values we have found in some intervals a good correlation (about 0.79) between the host CPU utilization metric and the host power consumption metric.

7.5.2 Analysis of Energy Mix Values

The correlation analysis has been also used to analyse CO_2 emissions [ICT4S2014]. As our goal is to deploy an application in a federated cloud environment, the evaluation and estimation of the emission factors is a very important step in our approach. As the factor may vary over time, it is important to know the value of the emission factor at the time when the application will run, so that the optimal deployment can occur.

There are different ways to assess the CO_2 emissions. There are electricity operators that periodically publish the 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 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). Besides the aggregated emission factors, some countries publish the real time energy mix via public web sites. For example, France energy mix can be retrieved through the information service *éCO2mix* available on the RTE website (http://www.rte-france.com/fr/) and data about the energy generation in UK are available through the BMRS (Balancing Mechanism Reporting System) website (http://www.bmreports.com/).

The availability of historical data can be exploited in order to identify regular and/or seasonal patterns that can be used in the deployment of applications. For example, a preliminary analysis of French data of January 2012 revealed the presence of a regular pattern of the emission factors during the week days and another pattern for the weekend days (see Figure 9).

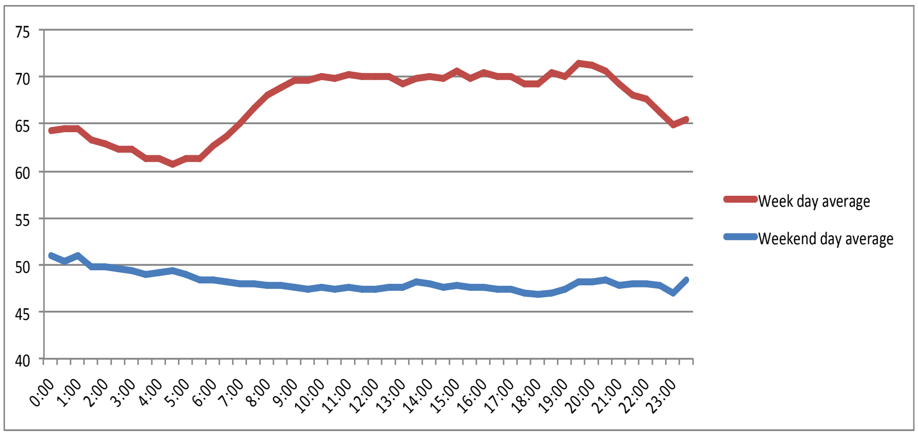


Figure 9 - Emission factors during the week and weekend days

The regularity of the emission factors that characterizes the weekdays has been proven by calculating the correlation indexes among the assessed values of the different week days.

Such trends are helpful in driving greener choices for the deployment of cloud applications. In fact, they can be exploited for making the user aware of the environmental impact of his/her application. In particular, two complementary approaches are proposed: (i) Immediate site selection, (ii) Execution shifting.

The *immediate site selection* approach selects the site where to deploy the application on the basis of the CO_2 emissions predicted for the future period that covers the execution time of the application. The selection algorithm works as follows:

1. Check the availability of the resources requested by the application on each of the sites S available obtaining a subset of sites S ′.
2. For each site in S′ predict the CO_2 emission due to the execution of the application given its estimated duration:
    a. If real time values are available for the site then estimate the pattern followed by the CO_2 given the past observation.
    b. If only an aggregated value is available for the site just multiply the value for the estimated duration of the application.
3. Compare the estimations and select the site S* with the lowest estimation.

The *Execution shifting* approach operates when the application can be postponed in time. In this case, the system can investigate better allocation that allows a greater reduction in CO_2 emissions. The user, in the deployment phase by using the ECO2Clouds portal, can specify the maximum allowed delay for the application execution. The execution shifting algorithm bases its behavior on the knowledge of the CO_2 patterns discussed above. Given this knowledge, for each site where the instantaneous energy mix is known, the algorithm works as follows:

1. for each site in S find the best execution starting point given the estimated duration of the application and the maximum allowed delay;
2. for each solution, analyze the resulting CO_2 emission values;
3. propose to the user a list of possible deployment solutions with their associated CO_2 emission estimation.

The user is involved in this process since they can decide which is the best solution for their needs by selecting an option from the ranked list.

For example, if we consider a scenario in which an application requires to be executed for 3 hours and we estimate an energy consumption of 3kWh. Let us refer to data collected for emissions in France (see Figure 9) and consider that a request arrives on Thursday 19th of January at 4:00 pm. The user specifies they accept to postpone the execution with a maximum delay of 48 hours. From an analysis of the trend, the execution shifting algorithm proposes several solutions to the user. The first solution is immediate deployment, with an estimated equivalent carbon emission of 209.7 gCO_2e. The second solution delays the execution by 10 hours, deploying the application on Friday 20th at 2:00 a.m. In this case, the estimated saving is 24.35 gCO2e. The last solution proposes the execution in the week end, starting at 7:00 a.m. of Saturday 21st, with a delay of 27 hours and an estimated saving of 66.5 gCO2e. The user can decide which solution is better according to their needs.

In Table 8 the three solutions are compared. The table reports both the estimated and the real values for CO_2 emissions for the three solutions. In the last column it is possible to see the saving in emissions that is obtained when delaying the application deployment. This value is obtained by comparing the effective emissions of the solution to the outcome of the immediate deployment. In this specific example, the algorithm can reduce the emissions by up to 30%.

Table 8 - Comparison of execution shifting outcomes

	Delay	Estimated gCO2e	Real gCO2e	Saving (%)
Solution 1	0	209.7	200.3	-
Solution 2	10h	185.4	167.1	16.6%
Solution 3	27h	143.2	140.3	30%

7.5.3 Analysis of the Influence of Application Deployment on Energy Consumption

The analysis of data extracted from the application executions can be also useful to understand the impact of the way in which the application has been deployed on the energy consumption [EuroECO2DC]. Such analysis is particularly useful to estimate the energy consumption at design time and understand the best configuration to use to deploy the application.

In particular, we use the Eels ECO$_2$Clouds use case and we analyzed different possible deployment strategies for its execution. Given N application instances, we want to find the best configuration, in terms of number of VMs needed to execute the application instances, execution policies (e.g., parallel or sequential execution), storage access strategies (e.g., synchronous or asynchronous), such that the best results concerning execution time and energy consumption are obtained. We will analyze the impact of different execution strategies with one or more virtual machines on response time, resources utilization, and energy consumption of the same configuration. In this specific scenario, we investigate five different deployment configurations that can be used to execute the Eels application. Each one of them is associated with a specific scenario that the users might encounter. The method proposed in this section can be also applied to other applications with a similar execution pattern.

The considered configurations are as follows:

Configuration 1 - synchronous parallel execution. In this configuration, several users execute the application to analyze the Eels migration behaviors (Figure

10). To minimize the user response time, several application instances, one for each user, are executed in parallel. A separate VM is assigned to each application in- stance; and all VMs are homogeneous (i.e., they have the same configuration) and synchronized in accessing the storage. While this configuration aims to achieve the shortest response time, it might exhibit a risk of resource contention since the storage can become the bottleneck of the system.

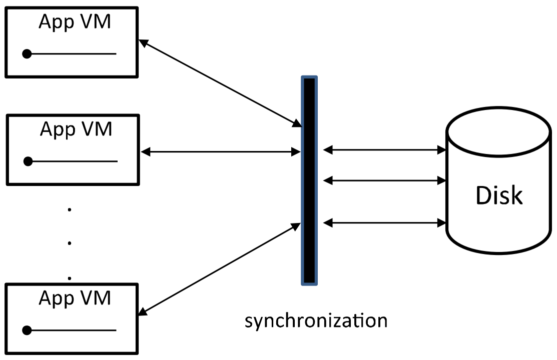


Figure 10 - Configuration 1: synchronous parallel execution

Configuration 2 - staggered parallel execution. This configuration is slightly different from the previous one with respect to the storage access (Figure 11). In this case, to avoid potential resource contention in data loading, storage accesses have been scheduled with a delay of 3 minutes, the required time to complete a data loading phase. The remaining part of the setup is similar to Configuration 1. At first sight, this configuration might lead to longer response time with respect to Configuration 1 due to the delay. However, by avoiding resource contention, it might result in shorter execution times when the number of VMs is large.

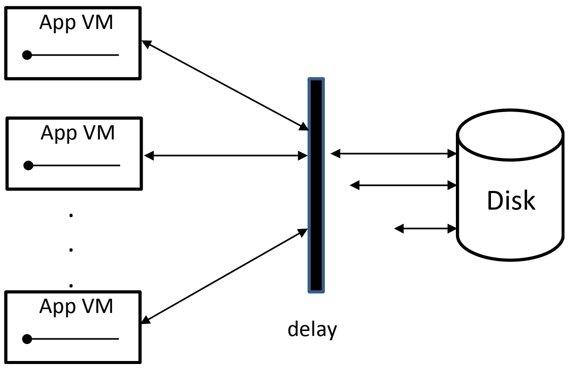


Figure 11 - Configuration 2: asynchronous parallel execution

Configuration 3 - sequential execution. This configuration describes the situation in which a user executes multiple instances of the application to analyze the same data set several times, possibly with different parameters (Figure 8). The time required to analyze the data is not a critical constraint while the minimum amount of resources used it is. Therefore, multiple application instances will be executed sequentially on the same VM and the accesses to the storage are sequential.

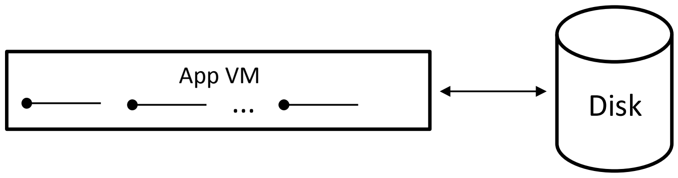


Figure 12 - Configuration 3: sequential execution

Configuration 4 - synchronous parallel execution with minimal resources. This configuration (Figure 13) is another deployment alternative to Configuration 1 that considers only the minimum amount of computational resources, i.e., one VM. Multiple application instances are deployed on the same VM, executed in parallel and storage accesses are synchronized. This configuration might result in longer response time due to the higher workload assigned to computational resources (i.e., the application VM). However, it will be interesting to compare its results with the ones of other configurations, analyzing the tradeoff of system response time and energy consumption, and evaluating the benefits in terms of energy consumption due to the limits of the computational resources.

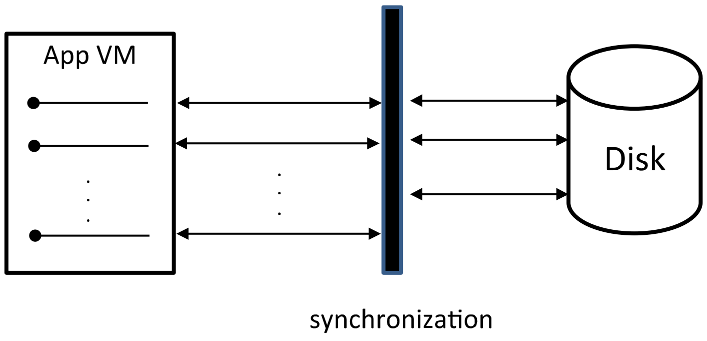


Figure 13 - Configuration 4 - synchronous parallel execution with minimum resources

Configuration 5 - staggered parallel execution with minimal resources. This deployment is alternative to Configuration 4, where we consider a delay for each application instance when accessing the storage (Figure 14).

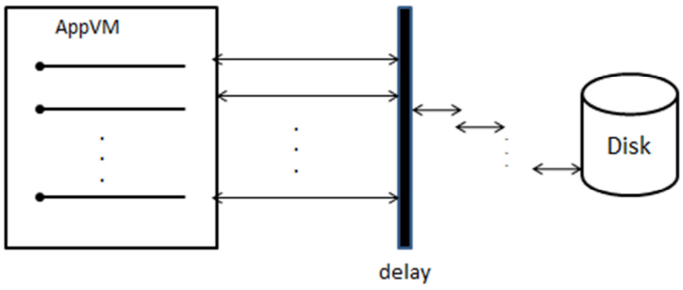


Figure 14 - Configuration 5 - asynchronous parallel execution with minimum resources

7.5.3.1 Power model

To compute the power consumption of the different configurations, it is necessary to adopt a power model able to predict the actual value of the consumption based on some runtime characteristics. Fan et al. (Fanetal.2007) describe a linear relationship between the CPU utilization and the total power consumption of a server. According to their model, the power consumption of a server *P(u)* grows linearly with the CPU utilization. The initial value is P_{idle}, i.e., the power consumption in the idle state, and the final value is P_{busy}, i.e., the power consumed at 100% of utilization. The following equation describes this relationship:

$$P(u) = P_{idle} + (P_{busy} - P_{idle}) \times U \qquad (1)$$

where U is actual value of the CPU utilization of the considered physical host.

While the presented equation computes the power consumption of a VM considering only one physical host, in our work we used eq. 2 to estimate the power consumption of an experiment *P(e)* when there are more than one VM involved and deployed on multiple physical hosts, assuming a single physical host allows to deploy up to a maximum number of VMs:

$$\mathrm{P(e) = P_{idle} \times ceil(N \div MaxVM) + (P_{busy} - P_{idle}) \times U \times N} \qquad (2)$$

where N is the number of VMs used in the experiment and MaxVM is the maximum number of VMs that can be deployed on a single physical host.

This model allows us to estimate the power consumption even in cases where multiple physical servers are used. With eq. 2 we estimate the power consumption for each configuration, using the P_{idle}, P_{busy} and U measured from our BonFIRE infrastructure. To estimate the total energy consumption of the experiment, we use eq. 3:

$$E(e) = P(e) \times R \qquad (3)$$

where E is the estimated total energy consumption, P is the estimated power consumption computed using eq. 2, and R is the system response time given by the models.

7.5.3.2 Deployment configuration analysis

We exploited the five different models presented above to extract useful insights of system performance and energy consumption. We compared five configurations in terms of system response time and energy consumption, with a number of application VMs ranging from 1 to 30. This comparison helped us to find the dominant and dominated configurations with respect to energy consumption and system performance (i.e. the response time). Figure 15 to Figure 18 show the comparison.

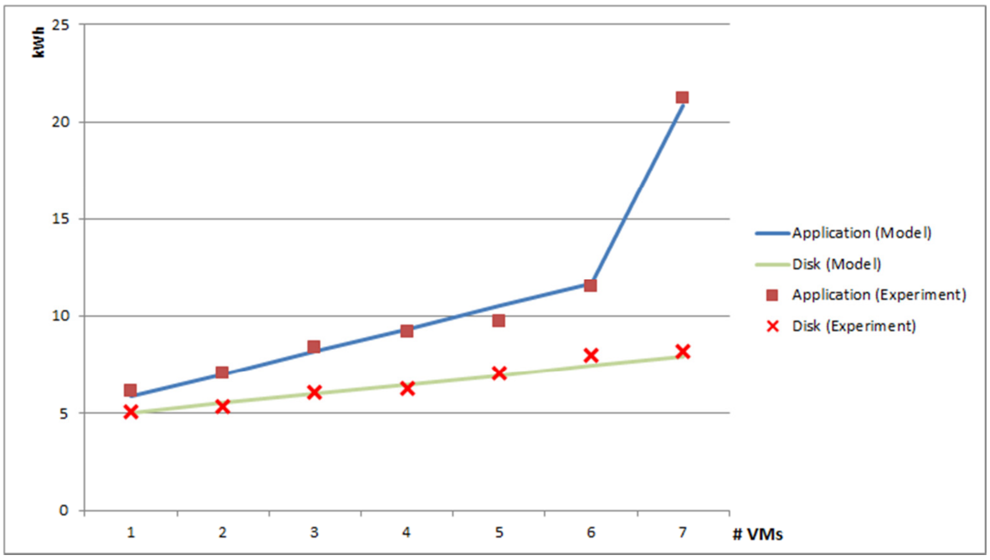


Figure 15 - Application and disk energy consumption for Configuration 1

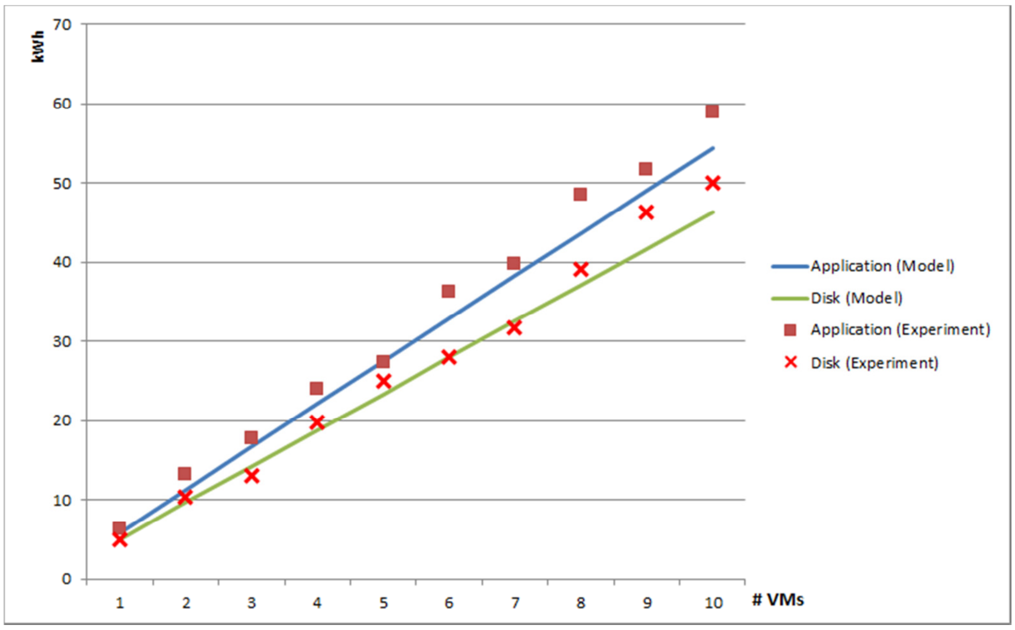


Figure 16 - Application and disk energy consumption for Configuration 4

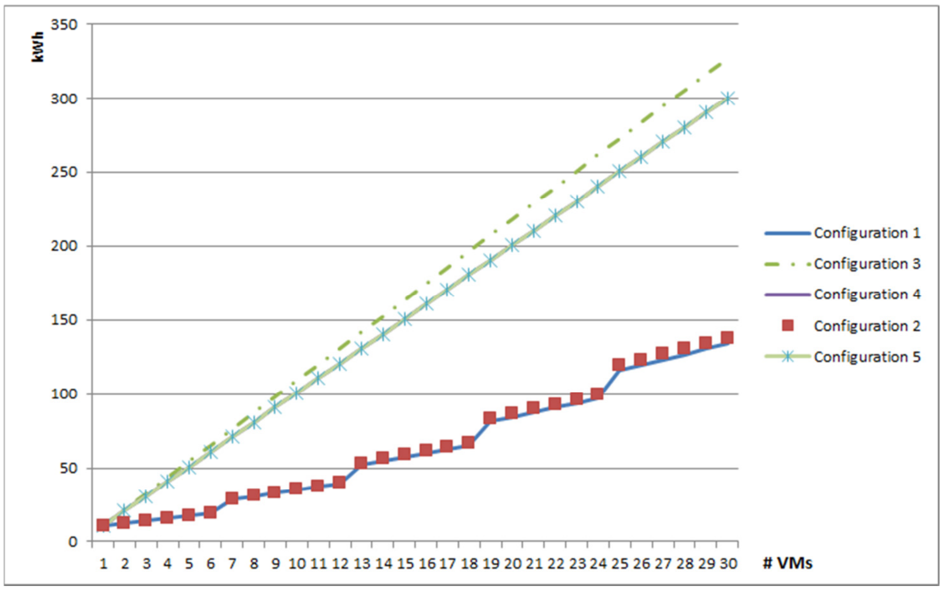


Figure 17 - Energy consumption comparison of different configurations

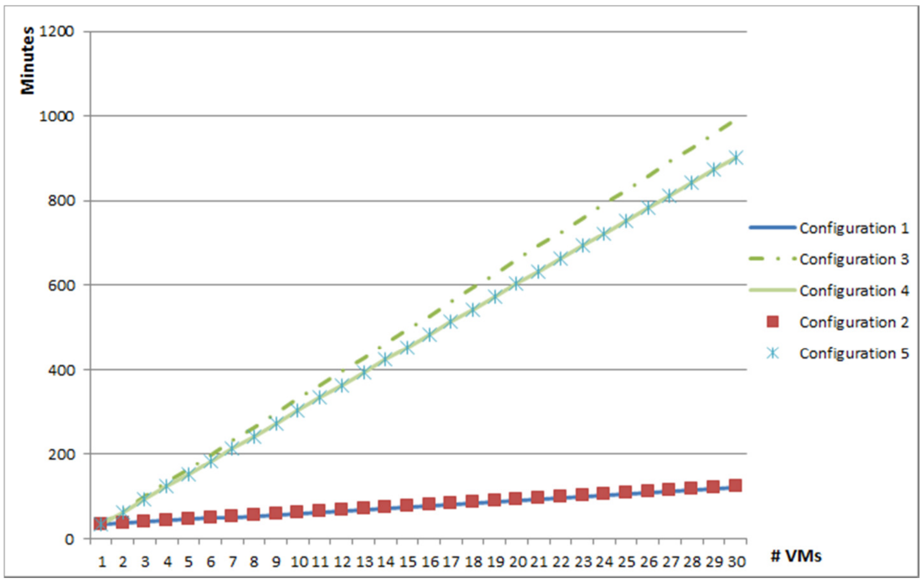


Figure 18 - System response time comparison of different configurations

Figure 15 – Figure 18 unveil a linear increase in energy consumption with respect to system response time in Configuration 3, 4 and 5; and a non-linear relationship of energy consumption in Configuration 1 and 2. The ladder step behavior in energy consumption of Configuration 1 and 2 is due to the ceil function (ceil(N ÷ MaxVM)) in eq. 2, which is related to the number of physical hosts required to host N VMs.

The figures also show that Configuration 3 is slightly dominated by the others, considering both energy consumption and system response time. Moreover, the energy consumption and system response time of Configuration 1 and 2 are identical, similarly for Configuration 4 and 5. This phenomenon can be explained by the way the delay is applied. In Configurations 1 and 4, the disk accesses are synchronized. These setups give the disk itself responsibility to schedule incoming requests; in fact, the delay is performed automatically by the disk. In

Configurations 2 and 5, the delay is performed manually by adding delay stations. So, for this type of problems, staggered disk accesses do not seem to provide a case for a better deployment configuration.

As a conclusion, assuming an unlimited number of resources is available, Configuration 1, with a single VM for each application instance, consumes less energy than using a single VM for all application instances, either executed in parallel or serial. Hence, Configuration 1 is the optimal deployment for this type of application profile. However, further analysis is needed to extend the results in cases in which congestion might occur at a certain number of application instances or in which data dependencies are present.

8 Conclusions

In D3.4 we presented an improved set of monitoring metrics required for the ECO$_2$Clouds project. This includes a detailed representation of the utilized metrics for three different monitoring layers, as well as showing which metrics were skipped because they contain trivial calculations that can be performed at another domain. Later we focused on the way the carbon footprint estimation is performed, which is the end goal of the project' monitoring system.

Also, in this deliverable we described an enhancement and implementation of the data mining approach. We have shown how the raw monitoring data is reduced in order to avoid disc space problem at the accounting host. The raw historical data are transferred to the storage host, where they are analysed to determine patterns, which can help develop strategies for the deployment of applications in the cloud.

In a next step we will present the results of WP3 as a white paper, focusing on more detailed results of the data analysis component of the Data Mining approach including good practices for cloud computing energy consumption optimisation issues. Also, in the future work in the ECO$_2$Clouds project we will use the presented monitoring infrastructure and the implemented metrics as basis for running experiments and adapting the use cases deployment in WP4 and WP5.

9 References

(D3.2 2013) Deliverable D3.2 "Design of an Efficient Monitoring Environment for Complex IT Infrastructures". March 2013.

(D3.3 2013) Deliverable D3.3 "Realisation of an enhanced Monitoring and Data Analysis environment (I)". June 2013.

(D2.4 2014) Deliverable D2.4 "Updated ECO$_2$Clouds Architecture ". February 2014.

(BonFIRE 2014) The BonFIRE Project Website, http://doc.bonfire-project.eu/R4.0.5/overview/infrastructure.html, Last visited: 10.06.2014.

(Zabbix) The Zabbix website documentation, https://www.zabbix.com, Last visited: 10.06.2014

(ICT4S2014) Cinzia Cappiello, Paco Melià, Barbara Pernici, Pierluigi Plebani and Monica Vitali "Sustainable choices for cloud applications: a focus on CO2 emissions" , Accepted for publication in the ICT4S conference

(EuroECO2DC) M. Gribaudo, N.T.T. Ho, B. Pernici, G. Serazzi, "Analysis of the influence of application deployment on energy consumption", accepted for publication in the EuroECO2DC

(Fanetal.2007) Xiaobo Fan, Wolf-Dietrich Weber, Luiz André Barroso: Power provisioning for a warehouse-sized computer. ISCA 2007: 13-23