An energy aware application controller for optimizing renewable energy consumption in Cloud computing data centres

Corentin Dupont, Mehdi Sheikhalishahi, Federico M. Facca
Create-Net
Trento, Italy
{*cdupont, msheikhalishahi, ffacca*}*@create-net.org*

Fabien Hermenier
Univ. Nice Sophia Antipolis
CNRS, I3S, UMR 7271, France
fabien.hermenier@unice.fr

***Abstract*—Sustainable energy sources such as renewable energies are replacing dirty sources of energy in order to address the environmental challenges of the century. In order to operate data centres with renewable energies we have to mitigate their volatile and variable nature. In this paper, we present the Energy Adaptive Software Controller (EASC), a generic software controller and interface that developers can use to make their application adaptive to renewable energy availability. Adaptivity is realized through the concept of working modes which allows to run an application under various performance levels. We advocate for a collaborative approach involving the developers of the applications in order to use the renewable energies more efficiently. The notion of EASC allows to abstract away the details of application scheduling, execution, and monitoring. We demonstrate the applicability and genericity of the EASC concept through four different instantiations. These instantiations cover two types of applications: task-oriented and service-oriented; and two kind of computing environments: Infrastructure-as-a-Service, and Platform-as-a-Service. The EASC has been trialled in the data centre of the healthcare agency of Trento, Italy and in the laboratory of HP Milan, Italy, with a mix of energy sources: national grid and local solar panels. The experimental results show how the EASC allowed to increase the renewable energies usage of 14% and 4.73% for Trento and HP Labs trials, respectively.**

***Keywords*-Renewable energy, Data centre, Adaptive application, Cloud computing**

I. Introduction

The rise of the energy consumption in data centres, and the high share of their electricity consumption around the world [1] induced data centres owners to take actions. Energy efficiency measures has been introduced in order to reduce the energy consumption of data centres, and now we move towards sustainable energy sources such as renewables [2] [3] [4] in order to address the current environmental challenges. For example, Google's data centres are currently operated with a 35% share of green energy[1], whereas 87% of energy consumption of Apple's data centres comes from renewables[2].

[1]http://www.google.com/green/energy/
[2]https://www.apple.com/environment/

With the recent adoption of renewable energies to power data centres [2], the research community enlarges its vision to associate with purely quantitative energy consumption reduction, the notion of quality of the energy consumed, i.e. the capacity to rely as much as possible on sustainable power sources. Differently from brown energy sources, the availability of renewable energies is very volatile and time dependent: e.g. solar power is obtainable only during the day, and is subject to variations due to the meteorological conditions. The goal is then to schedule the workload of running applications according to the forecasted renewable energy availability.

The problem is, however, that data centres are rather heterogeneous environments with many different kind of applications with different kind of computing styles: for example Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). The applications running on those platforms do not cooperate to provide a scheduling of the workload that matches the times where the renewable energies are available.

In order to address this problem, researchers proposed various solutions. For instance, in [5] authors proposed energy adaptive applications that reconfigure their runtime execution performance according to energy objectives, however not considering renewable energies. Authors in [2] [3] [4] have taken into account renewable energies; however, these papers are application-specific. For instance, with the aforementioned solutions service-oriented applications cannot get adaptive. The emergence of cloud computing provided the users with the ability to scale their services horizontally or vertically according to the demands [6]. In addition, the notion of adaptive applications has been introduced in [7].

In this paper, we advocate for a generic and a collaborative approach that involves developers of applications thanks to the notion of Energy Aware Software Controller (EASC). The EASC is a generic software controller that application developers/administrators can use to make their application adaptive to renewable energy availability. A data centre can support a mix of task-oriented and service-oriented applications that are made adaptive to renewable energy availability. The EASC abstracts away the details of application scheduling, execution, and monitoring. The EASC constitutes the southbound subsystem of a prototype implementation of the EU project DC4Cities[3].

This paper makes the following contributions:

- The EASC, a software controller allowing to develop renewable energy adaptive applications of any type. Software developers can integrate it into their applications to make them adaptive to the current energetic situation.
- An API, the *Energy Adaptive Software Control Interface*, allowing applications to receive energy related instructions.
- Four different EASC instantiations to demonstrate the genericity and the applicability of our concept. These instantiations cover two types of applications: task-oriented and service-oriented; and two kind of computing environments: IaaS and PaaS.
- An practical validation of the task-oriented and the service-oriented EASCs inside two data centres in Italy. The task-oriented EASC was trialled in the data centre of the healthcare agency of Trentino that is powered by the national grid, while the service oriented EASC was trialled in the lab of HP Milan that is also powered by local solar panels, in addition to the national grid. These experiments confirmed that the EASC concept allows to increase the renewable energy usage of both kinds of applications. The renewable energy percentage improvement are 14% and 4.73% for Trentino and HP Labs trials, respectively.

The remainder of the paper is organized as follows. Section II describes the EASC concept and architecture. Section III describes the various EASC instances. Section IV presents the experimental results. Section V is a review of prior work. Finally, Section VI describes next steps and concludes the paper.

II. Energy Aware Software Controller

A. Overview & Context

In this section, we describe an overview of the EASC and its context. For each application in a data centre, the EASC role is to build a workload scheduling plan according to a power budget, to enact the plan and finally to monitor its activities and energy consumption. Each EASC in the data centre receives a power budget from the central system of DC4Cities, called the CTRL. This is done through the interface called the *Energy Adaptive Software Control Interface*. This interface defines a standardized communication mean for all energy related information toward the applications of a data centre. The power budget transmitted consists in a recommended power consumption for each time of the day.

[3]http://www.dc4cities.eu/

To compute the power budget, the CTRL collects a certain number of informations from the data centre, from the energy provider and from the so-called Energy Management Authority (EMA). The EMA provides a set of objectives regarding the consumption of the renewable energy by the data centre. In the context of a Smart City, the EMA is integrated inside the city management operations to provide renewable energy consumption objectives for the major energy consumers through the city. The CTRL also needs the energy availability and source mix forecasts to determine its reference consumption levels to meet the energy and power goals provided by the EMA. This information is retrieved and continuously updated from the energy and power forecast data providers. After receiving the renewable energy availability and forecasts, the CTRL computes the ideal power quota for the next 24 hours for the full data centre. This ideal power quota is split into a set of power budgets for each application of the data centre, and transmitted to the EASCs through the *Energy Adaptive Software Control Interface*. This mechanism is applied iteratively; each application is monitored and if needed the CTRL will augment or reduce its allocated power budget through time.

B. Architecture

This section describes the EASC architecture, its components and APIs. Figure 1 illustrates a high-level architecture of the EASC components, including its API interface toward the CTRL. The EASC implements workload scheduling techniques that can reconfigure the applications according to the power budgets. The power budget is expressed as a series of time slots, covering a full day, with a power associated to each slot. Each slot denotes a recommended amount of power that the application should consume. We assume a time slot is a 15 minute period in our experimentation. The EASC selects the possible working modes (WMs) for the application according to the power budgets, the energy profiles of the working modes, and the SLAs.

The EASC allows the application operator to define the KPIs of his application, its SLA and its working modes. The KPIs are measured in term of a number of *business items* per unit of time. For example, the operator can define the business item to be the number of web pages served by his application, a number of files processed or a number of reports generated. The SLA are defined as a threshold on the KPIs that the EASC must guaranty. Depending on the type of application, the objectives of the SLA can be cumulative (task-oriented application) or instantaneous (service-oriented application). A working mode consists of:

- an actuator able to start and stop the working mode,
- an estimated power consumption,
- a maximum attainable performance for the KPI defined,
- a certain number of resources associated to that working mode.

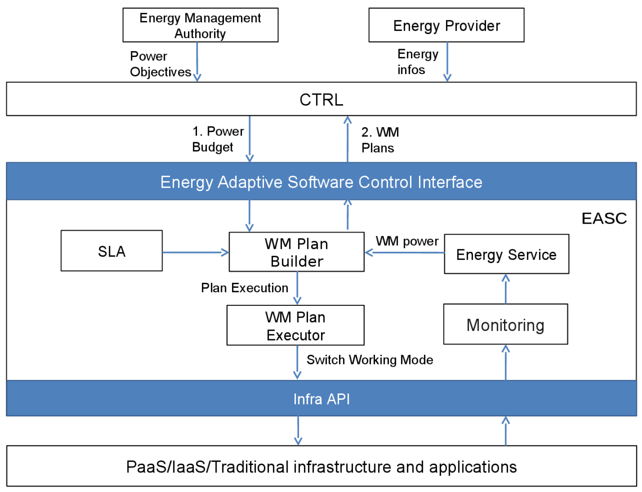


Figure 1: The EASC architecture

Based on this information, the EASC computes a scheduling for the application called the working mode plan. A working mode plan is composed of a working mode for each time slot. The EASC shall be able to reconfigure the application according to the working mode plan selected. For example, the EASC will send to the cloud manager reconfiguration actions in order to instantiate or stop VMs, boot or stop nodes. In addition, the EASC monitors the *work done* by its application in term of business items processed. We can then deduce a *work left to do*: it is the difference between the guaranteed number of business items (SLA) and the work done. This work left to do is what we have to schedule within the working mode plans. In terms of monitoring, the EASC shall be able to measure the KPI of the application at any moment. The EASC calls an so-called *Energy Service* to compute the expected energy profile for each working mode. This service allows to predict the consumption of the various components of the infrastructure (servers, VMs, containers). For each component of the data centre, a regression analysis is performed by the Energy Service on historical data in order to find a model of its consumption.

III. EASC INSTANTIATIONS

This section introduces the implementations of the EASC concept for various application types and environments.

A. EASC for Task Oriented Applications

Task oriented applications are characterized by the fact that they have a certain amount of tasks to perform. Those tasks are characterized by a minimum start time, a deadline, and a wall-time. Some of those tasks are deferrable: for example an anti-virus scan have to be performed every day, but can be scheduled at various time of the day. Other examples of task oriented applications include video conversion services or report generation applications.

When generating the working mode plan, the EASC can follow various policies:

- *Proportional*: this policy schedules the tasks in such a way that the expected energy consumption profile is similar to the power budget profile.
- *Aggressive*: this policy schedules all the tasks during the hours where there is the maximum renewable energy availability. In a typical day with solar energy available, the EASC will use the most powerful working modes during the central hours in order to complete the tasks. However, it is relatively risky; if the renewable energy forecast changes through time there is a risk that the SLA will not be met.
- *Eager*: this policy schedules all tasks at the earliest possible. This policy is more appealing when there is uncertainty over the future availability of renewable energy; therefore it is rather conservative.

The algorithm presented in Figure 2 allows to schedule the tasks of the application so as to follow the policies presented. It presents two parameters: the aggressiveness and the eagerness. The aggressiveness controls the possibility for the application to consume more or less aggressively the renewable energies. For example, at a high aggressivity level the application will run at the highest performance level when renewable energies are available, and at the lowest performance when they are not. On the other hand, the eagerness controls the necessity for an application to complete its tasks more or less early. A low eagerness allows to be more flexible with regard to the scheduling of the applications tasks during the period of availability of the renewable energies.

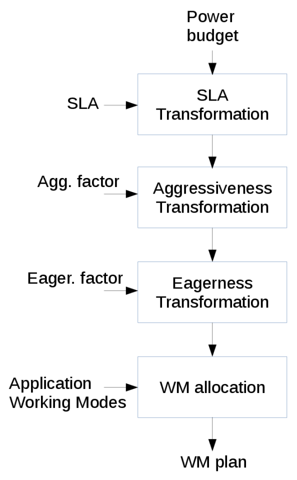


Figure 2: EagerAgg algorithm overview

In the algorithm 1, the function $AggTransform$ transforms a power budget by favoring the times of the day where there is more renewable energy. The resulting power consumption plan consumes the same amount of energy

Algorithm 1 EagerAgg algorithm

```
 1: constant workingModes : List of WorkingMode
 2: function AGGTRANSFORM($pb : Listof(TimeSlot, Pow), aggFact : Float$)
 3:     for all $(timeSlot, power) \in pb$ do
 4:         $power \leftarrow power * aggFact$
 5:         ▷ normFactor is computed so that the total energy of the final power budget remains the same than the initial one
 6:         $power \leftarrow power * normFactor$
 7:     end for
 8:     return $pb$
 9: end function
10: function EAGERTRANSFORM($pb : Listof(TimeSlot, Power), eagerFactor : Float$)
11:     $\Delta BP \leftarrow max(pb) - min(pb)$
12:     $eagerVector \leftarrow defEagerVector * \Delta BP * eagerFactor$
13:     ▷ sum of two vectors of equal length
14:     $eagerBP \leftarrow pb + normEagerVector$
15:     return $eagerBP$
16: end function
17: function SLATRANSFORM($pb : Listof(TimeSlot, Power), sla : Float$)
18:     ▷ normFactor is computed so that the total energy of the final power budget allows to run the correct working modes during enough time to cover the full SLA
19:     $normFactor \leftarrow f(workingModes.maxBizPerf, sla)$
20:     for all $(timeSlot, power) \in pb$ do
21:         $power \leftarrow power * normFactor$
22:     end for
23:     return $pb$
24: end function
25: function WMALLOCATION($pb : Listof(TimeSlot, Power)$)
26:     ▷ sort BP by power
27:     $sortedBP \leftarrow sort(pb))$
28:     for all $(timeSlot, power) \in pb$ do
29:         ▷ select the WM that have the power closest to the current power budget power
30:         $plan(timeSlot) \leftarrow nearest(workingModes, power)$
31:     end for
32:     return $plan$
33: end function
```

that the input power budget, but it presents more power in the maxima of the curve, and conversely less power in the minima of the curve. This allows the function WMAllocation to select more powerful working modes when renewable energy is available, and thus complete more quickly the tasks to perform. A high $aggFactor$ makes the application consumes the renewable energies more aggressively. The function $EagerTransform$, on the other hand, transforms a power budget by favouring the first time slots in the power budget. It uses a vector $defEagerVector$, which have the same size than the power budget. This vector $defEagerVector$ defines a simple descending array of powers, for example [100W, 99W .. -99W, -100W]. It represents the preference for the early time slots in the power budget. The transformation defined in $EagerTransform$ then transforms the power budget so as to allocate more power to the early time slots. The input factor $eagerFactor$ allows to tune this operation according to the preference of the application for running its tasks eagerly or not. Like for $AggTransform$, this transformation allows the function $WMAllocation$ to select more powerful working modes at early time slots, and thus complete earlier the tasks to perform. The $SLATransform$ ensures that the power budget has enough energy to allow the application to complete its full SLA. If it's not the case, a normalization factor is applied to the power budget to augment it or reduce it accordingly. The complete $EagerAgg$ algorithm is the simple composition of these four functions.

B. EASC for Service Oriented Applications

Service oriented applications, by contrast with the task oriented applications, do not define specific tasks with a start and a stop dates. On the contrary, they have to maintain a certain level of performance all the time, to serve clients. Typical examples of such applications include Web servers, database services and mail servers. The services are characterized by one or several KPIs together with their mandatory levels for each time-slot. For this type of application we created an algorithm called the *MinMaxAgg* presented in algorithm 3 and Figure 2. To allow a certain flexibility of the application, a system of reward/penalty is included in the SLA. In accordance with the client, if the application is running slightly under the SLA, a reward/discount is paid by the data centre.

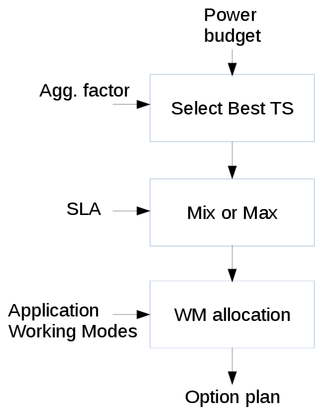


Figure 3: MinMaxAgg algorithm overview

In this algorithm, the function $BestTS$ selects the X timeslot of the day where there is more power in the power budget. Those timeslots are generally corresponding to the best hours for renewable energy availability. The factor X is computed relative to the aggressiveness factor. During those best times, a working mode allowing a better performance than requested by the SLA is selected. During the rest of the hours, a working mode slightly under the SLA is applied. Reward and penalty apply correspondingly, they are calculated so that they cancel each other.

C. EASC-PaaS

The EASC design has been extended to support Platform-as-Service based applications with EASC-PaaS. PaaS provides services to easily provision, scale, and monitor applications with a limited user/administrator intervention. In

Algorithm 2 MinMaxAgg algorithm

```
constant SLA: List of (TimeSlot, BusinessPerf)
constant workingModes : List of WorkingMode

function MINMAXAGG($pb : Listof(TimeSlot, Power), numTS : Int$)
    return $MinMax(BestTS(bp, numTS))$
end function

function MINMAX($filteredBP : Listof(TimeSlot, Power)$)
    ▷ for the best timeslots
    for all $(timeSlot, power) \in filteredBP$ do
        ▷ select the WM closest to the power budget, without violating the SLA
        $plan(timeSlot) \leftarrow getWM(SLA(timeslot).businessPerf, power$
    end for
    ▷ for the rest of the timeslots
    for all $(timeSlot, businessPerf) \in (SLA \setminus filteredBP)$ do
        ▷ select the WM that have the business performance just above to SLA
business performance
        $plan(timeSlot) \leftarrow above(workingModes, businessPerf)$
    end for
    return $plan$
end function

function BESTTS($pb : Listof(TimeSlot, Power), numTS : Int$)
    ▷ sort BP by power
    $sortedBP \leftarrow sort(pb))$
    ▷ get the numTS first timeslots
    $filteredBP \leftarrow take(sortedBP, numTS))$
    return $filteredBP$
end function

function GETWM($minBusinessPerf : Float, power : Power$)
    ▷ get only the WM that have a sufficient performance
    $filteredWM \leftarrow filter(workingModes, workingMode.businessPerf >$
$minBusinessPerf)$
    ▷ get the WM that have a power closest to the reference power
    $filteredWM2 \leftarrow nearest(filteredWM.power, power)$
    return $filteredWM2$
end function
```

EASC-PaaS the concept of working mode is mapped to the scaling operation services provided by PaaS infrastructures.

However in a traditional PaaS environment, scaling up and down an application will not necessarily have a big impact on energy consumption. The reason is that most PaaS architectures have static provisioning: scaling down an application or a group of applications will not result in the switching off of physical servers. In the Cloud Foundry[4] PaaS environment for example, a certain number of VMs[5] able to host application containers are provisioned when the infrastructure is installed, and does not change afterwards unless an operator redeploys the infrastructure manually. The applications are in turn embedded inside so-called *containers* hosted on the VMs.

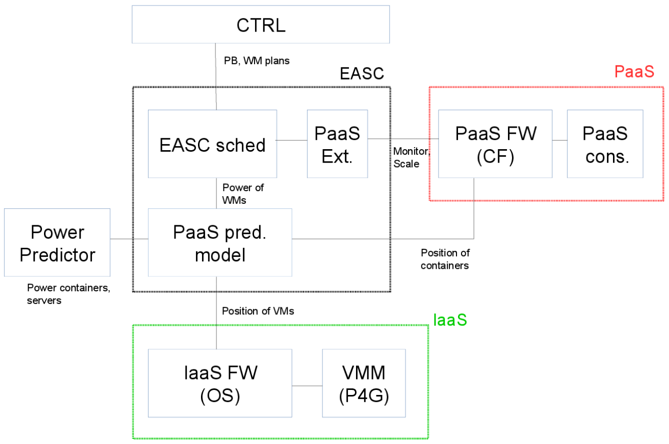
The energy management must then takes place in the three layers:

- *application layer:* The EASC will scaling up and down the number of containers owned by an application based on the renewable energy availability,
- *PaaS layer:* the containers must be consolidated inside the minimum number of VMs,
- *IaaS layer:* the VMs must be consolidated inside the minimum number of physical servers. The freed-up servers should then be switched off to save energy.

[4]https://docs.cloudfoundry.org

[5]https://docs.cloudfoundry.org/concepts/architecture/execution-agent.html

Figure 4: EASC-PaaS architecture



The Figure 4 provides a high-level view of the EASC-PaaS interactions with its environment. In this figure, the *EASC-Scheduler* module is the normal scheduler for task and service oriented applications described above. Based on this scheduling, the scaling of the application is performed through the *PaaS Extension* module, that connects to the PaaS infrastructure. The PaaS infrastructure is then instrumented to perform the consolidation of the containers in the VMs. This is done by an external component called the PaaS consolidator. This component is also in charge of augmenting or reducing the number of VMs as needed. Similarly, the IaaS layer is instrumented to allow VM consolidation on the servers. This is also performed by an external VM consolidator engine called Plug4Green [8]. The final objective of this tool-chain is to make sure that the scheduling operations performed by the EASC results effectively in energy consumption variation in the PaaS environment. Finally, the EASC-PaaS contains a component called *PaaS Prediction model* able to predict the behaviour of the underlying infrastructure and to compute the power consumption of each working mode accordingly, so as to permit an accurate scheduling.

D. EASC-IaaS

The EASC-IaaS aims at controlling the resource allocation of VMs running on nodes with regards to an energy budget and different SLAs. Similarly to existing IaaS platforms, the resources that are subject to adaptation are the computational resources. In practice, the EASC increases or decreases the computational power allocated to the VM virtual CPUs. As the CPUs are the hardware components that consumes most of the power inside a server, this adaptation permits to align the VM computational performance with a power budget.

To abstract the computational power from the hardware peculiarities, we evaluate computational power using a metric called CU (Computational Unit). This approach is aligned with current practices. For example, Google Compute Engine uses GCEUs to define the compute power of their templates[6].

A SLA for the EASC-IaaS is defined as a minimal amount of CU to allocate to a VM over the agreement period while a working mode consists in a particular CU allocation, so a particular performance level and power consumption. Scheduling the working modes for an EASC-IaaS consists then in choosing for each time slot, a working mode that is below the power budget while ensuring, that at the end of the agreement, the total amount of allocated CU will be at least equals to the threshold stated in the SLA.

To enact a working mode, the EASC-IaaS sends a CU allocation command to the underlying IaaS. In the case of an OpenStack IaaS, this CU allocation command will the the CPU entitlement feature[7] to cap the computational power of the VMs.

IV. Experimentations and Evaluation

In this section, we present the experimentations within two trial sites in Trento and Milan. We have measured *RenPercent* metric in order to evaluate the proposed solution in terms of renewable energy usage. *RenPercent* expresses the percentage of renewable energy consumed by a DC in a certain period of time (1 day, 1 week, etc.). This metric is the main metric to assess the achievement of the EASC. With this metric we can compare the coverage of the DC energy consumption of renewable energy before and after implementing the EASC.

A. Trento Trial

In this trial, an EASC encapsulates a real application producing medical reports, within the data centre of the healthcare agency of the Trentino province.

1) Application Specification: Trento trial application is a compute-intensive task-oriented application. Users need to wait some minutes to get the report generation done. The report generation process can be scheduled in advance; thus, the idea is to prepare a cached copy of each report and have it ready once the user asks for it. This leaves an opportunity for EASC to shift the report generation in time during a day. The only constraint (as per SLA) that EASC planning has to respect is that 780 reports have to be generated within a day, from midnight to midnight.

The application business unit for this trial is the *Report* and the business performance (BizPerf) unit is the number of reports generated per minute. Table I defines five working modes based on the number of parallel processes that run simultaneously to generate reports; *minimum* corresponds to execution of one process at a time (sequentially); *medium1* corresponds to execution of 4 parallel processes, etc. *ServersOff* working mode was defined for referring to the case of no activity execution (main production servers off). The baseline has been chosen with the application producing reports continuously until the SLA is reached, at a low level of performance using the working mode *minimum*.

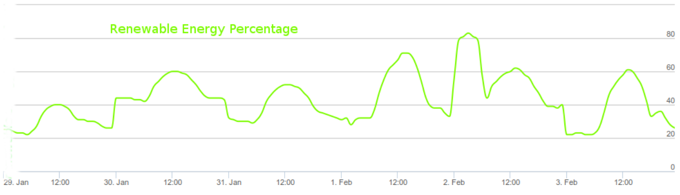
[6]https://cloud.google.com/compute/docs/machine-types#gceu
[7]https://wiki.openstack.org/wiki/CPUEntitlement

Table I: Trento Trial Card

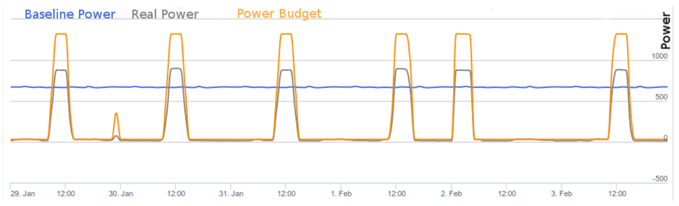
WM	Processes	BizPerf	Power (Watt)
ServersOff	0	0	19,5
minimum	1	0,53	665,8
medium1	4	1,88	761,9
medium2	7	3,0	844,4
maximum	10	3,33	864,2

In this trial, hardware resources include two IBM rack servers as compute nodes to run trial application: IBM xSeries 366 8863-3RG: 4 Intel Xeon Processor 7020 (2M Cache, 2.66 GHz, 667 MHz FSB) 4 cores, RAM 32 GB, HDD 30GB-RAID1, HDD 200GB-RAID5.

2) Energy Mix: The trial has a single power source from the Italian national grid. We followed a Measurement & Verification (M&V) methodology [9] and built six days profile that reflects typical days in various seasons in Italy. This methodology allowed us to simulate an entire year trial run with just 6 different profiles, one per day.



(a) Renewable energy percentage in national grid



(b) Power actual vs power baseline vs power planned

Figure 5: Trento trial execution behaviour with DC4Cities

3) Evaluation: Figure 5a shows the renewable energy percentage in the Italian grid for the six profiles consecutively. Figure 5b represents power budget, real power consumption, and baseline power consumption. This graph show that the baseline power (blue curve) is flat as expected, it also shows that the real power (grey curve) follows closely the power budget (orange curve). The peaks in the renewable energy percentage are always accompanied by peaks in the power budget and real power curves. Comparing Figures 5a and 5b we can observe that the EASC is able to follow the renewable energy percentage curve. In sum, the graphs shows that the EASC tries to exploit the peaks in renewable energy by switching to powerful working modes in order to perform all the tasks in the shortest time possible.

Table II presents the numerical results of RenPercent metric for all six days. In addition, this table presents average results of all six days. The last row presents results for a simulation of the entire year (six days spread on an entire year).

Table II: Trento Trial Results

Profile	RenPercent	RenPercent baseline
Day 1	37.65%	30.03%
Day 2	58.30%	49.24%
Day 3	49.73%	39.19%
Day 4	66.56%	45.98%
Day 5	77.94%	57.27%
Day 6	56.29%	37.06%
All days	57.88%	43.13%
A year	48.22%	37.39%

Considering all six days together the renewable energy percentage went from 43.1% up to 57.9%, an absolute improvement of 14%. The renewable energy usage improvement is greater than 10% for all days. Moreover, the renewable energy usage is very close to the maximum percentage of renewable energy available in the grid; this difference is always less than 5%.

B. Milan Trial

In this section, we describe Milan trial that presents a service-based application. The experiment takes place in HP Italy Innovation Center lab-grade resources hosted at HP premises in Milan, Italy.

1) Application Specification: HP trial workload is based on a Web application called HP Life. It is a testing lab for a worldwide service offered by HP. This application needs to be always available to the users (24 hours per 7 days) as it offers a Web based e-learning platform for entrepreneurs spread over the globe. Due to this type of SLA, this trial is quite different from the Trento trial; furthermore the workload cannot be shifted in time. The KPI for this service is the total number of requests served per minute (Requests/minute). There is a specific SLA value for each timeslot of the day expressing the expected business performance during that time interval. The red curve in Figure 6 presents the SLA. The other coloured curves represent the trial performance for each day.

In this trial, a working mode can span over distributed data centre resources around the world, i.e. Region US West, Region US East, Region Europe. Working modes definition are aligned to the concept of active region (data centres sites) balancing the workload of the system:

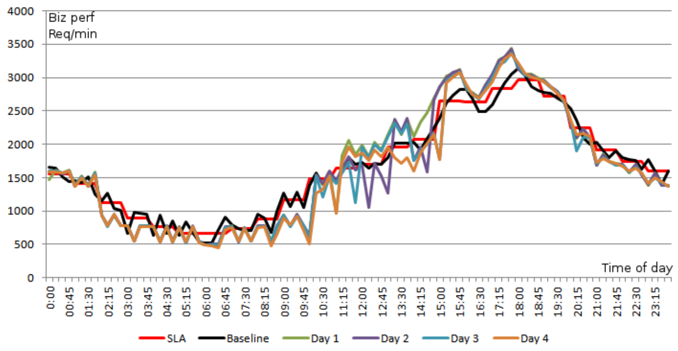


Figure 6: SLA and measured performance for each day

- WM3SF: All 3 sites work at full capacity (WM-ID: 30).
- WM2SF1SM: 2 sites work at full capacity, 1 site works at minimum capacity (WM-ID: 25).
- WM2SF: 2 sites work at full capacity (WM-ID: 20).
- WM1SF1SM: 1 site works at full capacity, 1 site works at minimum capacity (WM-ID: 15).
- WM1SF: 1 site works at full capacity (WM-ID: 10).
- WM1SM: 1 site works at minimum capacity (WM-ID: 5).

WM-ID is proportional to the amount of used resources and corresponds to the working mode power. The WM-ID are numerical values when displaying working modes in the daily graphs. Figure 7 represents the overall power consumption of the system when running in a certain working mode (on the Y axis) while delivering a certain business performance (on the X axis). It is visible on this graph that each successive working mode allows to reach a bigger business performance, but at a bigger energetic cost overall: the curve for a given working mode is practically always above the one of the previous working mode.

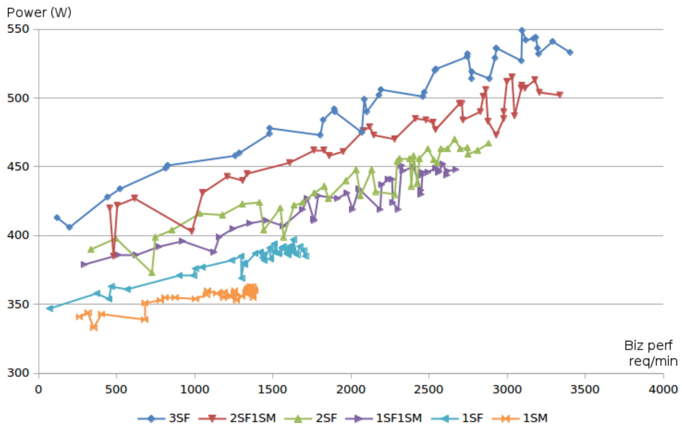


Figure 7: Business Performance (Req/min) versus Power Consumption (W) for each working mode

The baseline has been measured on default system configuration, i.e. when all servers are turned on and operational (a trial run with working mode WM3SF).

2) Energy Mix: The HP experiment has a dual power source: one from Italian national grid, and the other from local renewable generation. On top of the roof of the Lab premises, 4 dedicated photo-voltaic (PV) panels have been installed. Each panel can provide a maximum power of 250W; thus with 4 PVs configuration, maximum power generation is 1KW. This energy context offers a unique combination of energy sourcing.

Following M&V methodology [9] we've got 4 day profiles for the PV production, and with its aggregation with the Italian national grid we built four days profile that reflect a full-year behaviour of energy ecosystem for HP Experiment in Milan.

3) Evaluation: HP trial has been run for each 4 day profiles once with the baseline, and once with the EASC.

Table III: HP Experiment Trial Results.

Profile	RenPercent baseline	RenPercent EASC	RenPercent Delta	RenPercent Delta%
Day 1	68.20%	70.34%	2.14	3.13%
Day 2	61.85%	64.68%	2.83	4.57%
Day 3	58.57%	61.65%	3.08	5.25%
Day 4	53.99%	57.26%	2.27	6.05%
Trial Days	60.65%	63.52%	2.87	4.73%

Figure 8 represents an in depth view of trial run for each day. The X axis is the hour of the day; the yellow area represents the PV power (primary Y axis is power in Watt) while the light green area is the renewable percentage of the grid (secondary Y axis on the right). The light blue line is the planned power (power budget in terms of EASC) and the red line is the actual power. The purple line in the bottom part is just a qualitative indication of the selected working mode (numerical WM-ID), ranging from 5 (1SM) to 30 (3SF) in steps of 5 units (see previous WM-ID table).

Similar to the Trento trial, the graphs in Figures 6 and 8 show that the working mode power, and business performance curves follow the planned power curve (power budget). The peaks in the light blue line (the planned power) are always accompanied by peaks in the purple line (working mode power), and the red line (actual power). In all, the graphs demonstrate how the EASC tries to exploit the peaks in renewable energy by switching to powerful working modes, and vice versa to lower performance working modes when there are less power budget while still respecting the SLA (the red curve in Figure 6).

Table III presents the numerical results for RenPercent metric. The average improvement for RenPercent metric is 4.73%. We observe that the improvement is less than the Trento trial. This is due to the demanding SLA of HP trial, and being a service-based application.

Table IV: HP Experiment Trial: power and performance comparison

Profile	Total Power Consumption (KWh)	Total BizPerf (requests served)	Efficiency (Requests/W)
SLA		479K (ref)	
Baseline	10.91 (ref)	480K (100.0%)	43.98 (ref)
Day 1	9.42 (-13.60%)	484K (101.0%)	51.40 (+16.87%)
Day 2	9.28 (-13.14%)	474K (98.87%)	51.14 (+16.28%)
Day 3	9.46 (-13.24%)	471K (98.25%)	49.80 (+13.23%)
Day 4	9.43 (-13.51%)	466K (97.22%)	49.43 (+12.29%)

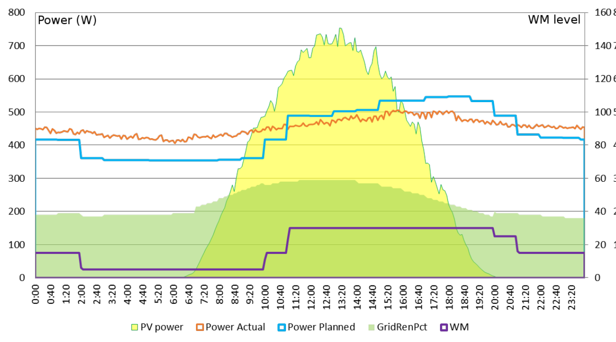
Nevertheless, Table IV summarizes the total power consumption, the total amount of work done (requests served during the day), and the "requests served/energy consumed" for each day of the trial and for the baseline. In terms of energy efficiency (second column in the table), this trial presented a significant amount of energy saving (over 13% energy consumption reduction). In addition, the Figure 6 illustrates that EASC optimization is not causing significant violations of the SLAs during the trials. That is the total work done (BizPerf) ranges between +1% to -2.78% with respect to SLA, as numerical results present in the Table IV (third column). This demonstrates that work done is not significantly affected. Therefore, the efficiency of the whole application is definitely increasing. The improvement in terms of BizPerf per Power, i.e. Work done/Watt, ranges between 12.29% and 16.87% (fourth column).

V. Related Work

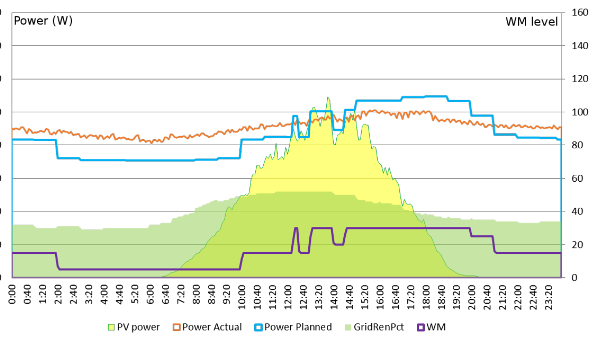
A survey of the literature shows many researches addressing high energy consumption of data centres and energy adaptive approaches in applications, for example [10] [2] [4] [3] and [11].

iSwitch [4] adapts the infrastructure itself. Servers are either powered through the grid or a renewable source. VMs are deployed by default on servers powered by renewable energy. When this source becomes scarce, the workload is migrated to servers powered by the grid and the idle servers are turned off. In order to adapt to renewables energy availability, in [12] the authors argue for either pausing VM executions or migrating VMs between sites based on local and remote energy availability. These work are similar to EASC-IaaS.

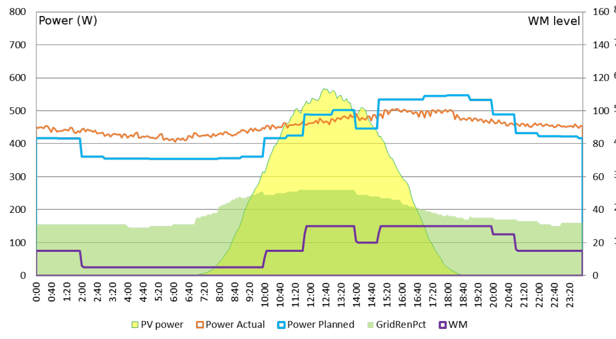
In [2] and [3] the authors adapt the workload to renewables when it is composed by deferrable jobs. According to forecasts, the jobs are delayed to periods where renewable sources are available. The work in [10] makes a step forward by adapting map/reduce applications. The number of workers and the number of servers hosting them is adapted according to the amount of available energy. In [3], Goiri et al. proposed GreenSlot, as a solar power-sensitive scheduling



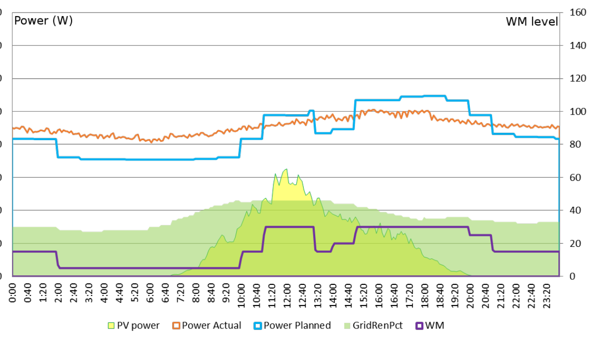
(a) Day 1



(b) Day 2



(c) Day 3



(d) Day 4

Figure 8: HP trial execution behaviour with the EASC

algorithm for data centre workloads. GreenSlot was shown to reduce data centre costs and increase green power consumption. In [11], authors exploited workload shifting in combination with local generation as an adaptation technique for two purposes: to avoid the coincident peak, and to save energy cost. In all these papers, workloads are task-based applications.

In addition, there are similar approaches to minimize energy costs, and carbon emissions. In order to reduce electricity costs in SCs, power-aware resource management without degrading utilization has been proposed in [13], [14]. The novelty of the proposed job scheduling mechanism is its ability to take the variation in electricity price into consideration as a means to make better decisions about job start times. The GreenStar Network [15] provides similar efforts toward developing green load-following carbon protocol. In [16], authors explored the opportunities for HPC clusters to adapt to dynamic electrical prices, variation in carbon intensity within an electrical grid, and the availability of local renewables. The results were that adaptation to the renewables availability and dynamic pricing lead to significant gain, while not for adaptation to the variation in the electrical grid carbon intensity.

By comparison with aforementioned approaches and solutions, the EASC approach allows to bring together different environments with a mix of deferrable (task-oriented) and non-deferrable (service-oriented) applications, running over IaaS or PaaS paradigms. In EASC we advocate for a generic approach that involves the infrastructure and the resource manager as well as the developers of applications.

VI. Conclusion and Future Work

The recent adoption of renewable energies to power data centres brings new challenges. While strong efforts are continuously made to reduce the energy consumption, the intermittent nature of the renewable sources impose also to align the performance of the applications with the energy availability periods.

In this paper, we presented the notion of Energy Adaptive Software Controller (EASC), a generic software controller that developers can use to make their application adaptive to renewable energy availability. To integrate the notion of EASC into a legacy application, a developer only needs to identify its various KPIs, working modes and to declare the commands to use to enact a working mode. The EASC then provides different scheduling algorithms to continuously choose the most appropriate working mode to use for the controlled application with regards to its SLAs and a power budget.

We validated the portability of the notion of EASC through the complete implementation of two legacy applications. The first one is a task-oriented application used to generate reports in the healthcare domain. The second one is HP Life, an international eLearning lab available through a Web application. We also confirmed the practical benefits of the developed EASCs on two different testbeds: one only powered by the Italian national power grid, and the other that is the case of HP life, through a dual power sources from photovoltaic arrays, and national power grid. Experiments over a week also proved the EASCs increased the usage of renewable energies by aligning the application performance with the energy availability periods.

As a future work, we plan to propose a coordination mechanism, at the data centres level, to decide for each EASC the most appropriate working mode depending on the energy availability and context-specific pricing models. We are also planning to enhance our model to support co-dependent EASCs that can perform adaptation at multiple levels.

Acknowledgments

This work has been carried out within the European Project DC4Cities (FP7-ICT-2013.6.2).

References

[1] Jonathan Koomey. Growth in data center electricity use 2005 to 2010. Technical report, Analytics Press, July 2011.

[2] Íñigo Goiri, William Katsak, Kien Le, Thu D. Nguyen, and Ricardo Bianchini. Parasol and GreenSwitch: Managing Datacenters Powered by Renewable Energy. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 51–64, New York, NY, USA, 2013. ACM.

[3] Íñigo Goiri, Ryan Beauchea, Kien Le, Thu D. Nguyen, Md. E. Haque, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. GreenSlot: scheduling energy consumption in green datacenters. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 20:1–20:11, New York, NY, USA, 2011. ACM.

[4] Chao Li, A. Qouneh, and Tao Li. iSwitch: Coordinating and optimizing renewable energy powered server clusters. In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, pages 512–523, June 2012.

[5] Frederico Alvares de Oliveira, Jr. and Thomas Ledoux. Self-management of cloud applications and infrastructure for energy optimization. *SIGOPS Oper. Syst. Rev.*, 46(2):10–18, July 2012.

[6] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, February 2009.

[7] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.

[8] Corentin Dupont, Fabien Hermenier, Thomas Schulze, Robert Basmadjian, Andrey Somov, and Giovanni Giuliani. Plug4green: A flexible energy-aware vm manager to fit data centre particularities. *Ad Hoc Networks*, pages 505–519, 2014.

[9] D6.2 - report on the experimentation phase1. evaluation report on the first trial cycle. Technical report, DC4Cities project, 2015.

[10] Íñigo Goiri, Kien Le, Thu D. Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. GreenHadoop: Leveraging Green Energy in Data-processing Frameworks. In *Proceedings of the 7th ACM European Conference on Computer Systems*, pages 57–70, New York, NY, USA, 2012. ACM.

[11] Zhenhua Liu, Adam Wierman, Yuan Chen, Benjamin Razon, and Niangjun Chen. Data center demand response: avoiding the coincident peak via workload shifting and local generation. pages 341–342, New York, NY, USA, 2013. ACM.

[12] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W. Moore, and Andy Hopper. Free Lunch: Exploiting Renewable Energy for Computing. In *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems*, pages 17–17, Berkeley, CA, USA, 2011. USENIX Association.

[13] Xu Yang, Zhou Zhou, Sean Wallace, Zhiling Lan, Wei Tang, Susan Coghlan, and Michael E. Papka. Integrating Dynamic Pricing of Electricity into Energy Aware Scheduling for HPC Systems. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 60:1–60:11, New York, NY, USA, 2013. ACM.

[14] Zhou Zhou, Zhiling Lan, Wei Tang, and Narayan Desai. Reducing Energy Costs for IBM Blue Gene/P via Power-Aware Job Scheduling. October 2013.

[15] Greenstar network: Building a zero-carbon network, 2015.

[16] D. Aikema and R. Simmonds. Electrical cost savings and clean energy usage potential for HPC workloads. In *2011 IEEE International Symposium on Sustainable Systems and Technology (ISSST)*, pages 1–6, 2011.