Minimizing energy and makespan concurrently in Cloud Computing workloads using Multi-Objective Differential Evolution

Daniel Balouek-Thomert*†, Arya K. Bhattacharya ‡, Eddy Caron†, Karunakar Gadireddy ‡, Laurent Lefèvre†
*NewGeneration-SR, Paris, France
†Inria Avalon team, LIP Laboratory, UMR CNRS - ENS de Lyon - INRIA - UCB Lyon 5668
University of Lyon, France
‡School of Engineering, Mahindra Ecole Centrale
Hyderabad, India
{daniel.balouek-thomert, eddy.caron, laurent.lefevre}@ens-lyon.fr, {arya.bhattacharya, karunakar14052}@mechyd.ac.in

***Abstract*—As the demand for Cloud infrastructures increases dramatically across the globe, the reduction of energy consumption in such infrastructures has become a challenge, urging for solutions that concurrently mitigate the environmental impact while maximizing performance benefits. Several approaches to reduce the power consumption of data centers have been described in literature. However, these approaches do not always provide means for providers (addicted to low energy consumption) and users (addicted to powerful resources) to specify how they want to explore such trade-off through novel methods of workload placement. In this context, we intend to perform an efficient selection of resources with respect to a certain quality of service and energy consumption. Our work synergizes two state-of-the-art technologies by combining Multi-Objective Evolutionary Algorithms (MOEA) with trade-off mechanisms using the DIET toolkit. We apply middleware-level mechanisms to provide a framework for assigning jobs to distributed resources and explore energy efficient resource provisioning without detailed knowledge of the underlying hardware platform. Evaluation of the proposed solution under different scheduling policies shows significant gains of energy consumption with some improvement on the overall workflow completion time. Experimental validation is performed on a real-life testbed using Cloud traces. We also evaluate performance of the optimization engine and scheduling overheads.**

Keywords*—Distributed computing, energy-efficiency, workload placement, resource provisioning, evolutionnary algorithm, middleware, workflow execution, multi-objective scheduling

I. Introduction

Many of the IT services that organizations utilize nowadays, depend on large computing infrastructures that are hosted either locally or at remote data centers [1]. A popular business model for renting out resources of a data center is provided by Cloud Computing, which enables customers to allocate computing, storage and network capacity over the Internet and pay by the hour of use. In recent years, concerns about energy consumption are increasingly becoming common as Clouds often consume a large amount of electricity to power and cool datacenters they rely on [2]. This situation is partially caused by an overprovisioning to ensure service delivery at peak hours, leading to underutilized resources in other times [3], [4]. Efficient allocation of tasks to resources can improve consolidation on a minimum number of nodes, while transitioning remaining unused nodes to low-power modes or shutdown [5], [6]. The implementation of the task-to-resource allocation policy consists in picking in real time at Cloud providers end the best combination of resources, in order to fit the customer's needs at lowest cost, risk and lowest energy consumption. Server allocation policies usually involves two actors: the Cloud provider who defines placement policies according to available resources while the customer submit sets of tasks to be executed. Such policies must benefit both the provider and customer in terms of costs and completion time.

In previous work [7], the present authors proposed methods for provisioning resources and distributing requests with the objective of meeting performance requirements while reducing energy consumption. *GreenPerf*, a hybrid metric, was introduced as a ratio of performance and power consumption for energy efficiency. The proposed solution considered willingness to perform energy savings by balancing user's and provider's preferences when scheduling the requests over the physical nodes. However, due to the contradictory nature of those objectives, *GreenPerf* did not explore the large possible range of solutions. The search and computation of these solutions is a NP-Hard problem, which can be formulated as an optimization problem with multiple contrary objectives: minimizing both energy consumption and completion time. In this work, we have used Non-Dominated Sorting Differential Evolution to obtain the best Pareto front with a spectrum of solution representing minimum energy at one end of the front and minimum makespan (completion time) at the other.

Since the first attempt to solve multi-objective optimization problems by using Genetic Algorithms [8], the field of Multi-Objective Evolutionary Algorithms (MOEAs) has almost continuously seen significant development and is widely used today in numerous applications. Among the algorithms developed in early stages, a few such as Vector Evaluated Genetic Algorithms [8], Niched Pareto Genetic Algorithms [9], Pareto Archived Evolution Strategy [10], Strength Pareto Evolution Algorithm [11], and the Non-dominated Searching Genetic Algorithm [12] are noteworthy in the sense that they triggered alternate lines of thought and were subjected to comparisons across multiple test cases. Zitzler [13] and Deb [14] carried out significant improvements in their earlier approaches [11], [12] resulting in SPEA2 and NSGA-II. Some aspects of the latter approach have been incorporated in the development of the Non-Dominated Sorting Differential Evolution II technique (NSDE-II), used in the current paper. Differential Evolution (DE) was selected as the evolutionary method of choice on the basis of the author's prior studies on the relative efficiency and merits of this against Genetic Algorithms, as reported in [15].

This work focuses on workflow applications that consist of multiple components (tasks) related by precedence constraints that usually follow the data flow between them, *i.e.,* data files generated by one task are needed to start another task. Although this is the most common situation, precedence constraints may exist for other reasons, and be arbitrarily defined by the user. We intend to integrate NSDE-2 as a Multi-Objective Optimization engine within a large scale infrastructure. NSDE-2 would be accessible as a remote service that accepts a workflow as an input and computes a set of placement solutions that minimizes energy consumption and performance requirements as an output. This output is to be placed and executed on the infrastructure using the DIET Middleware.

This paper introduces several contributions: (i) an evolutionary approach to workflow placement based on our previous work, (ii) a choice of solutions to the user based on his priorities, ranging from best-energy to best-makespan, and intermediates and (iii) an experimental protocol using a real-life testbed and Cloud traces.

The remainder of this paper is structured as follows. Section II presents Differential Evolution, the DIET toolkit and a short summary of related works from the literature. In Section III, we introduce the Non-Dominated Sorting Differential Evolution II algorithm. In Section IV we present the problem formulation. In Section V, we propose a generic and customizable infrastructure for workload placement on a large scale infrastructure. To assert the pertinence of our algorithm, an experimental protocol is defined in Section VI. Finally we conclude in Section VII and give insights for perspectives and future improvements of our approach.

II. Background and Related work

In this section we provide an overview of evolutionary optimization. We then introduce the DIET middleware and the features used in this paper. Finally, we present a short summary of related work on workload placement and energy efficiency.

A. Differential Evolution

The developments in Multi-Objective Evolutionary Algorithms refered in Section I have been along the track of Genetic Algorithm (GA) [16]. At the basic algorithm level, Differential Evolution (DE) was formulated as an alternate approach to GA by Storn and Price [17], [18]. About the same time, Particle Swarm Optimization (PSO) was proposed by Kennedy and Eberhart [19], and Ant-Colony Optimization by Dorigo et al [20] [21]. Though the latter two are not evolutionary algorithms per se, they are also expressions of natural life processes in the optimization domain and can be broadly classified as belonging to an extended family of evolutionary optimization techniques. The present authors have applied DE in a few complex industrial processes [15], [22]; the latter work also provides a comparison in computational efficiency for that industrial process between GA and DE demonstrating that DE comes out favourably. In [23], the authors have systematically compared DE with PSO and concluded that DE performs better. Due to the above comparisons and the authors' prior experience in the field they have used DE as the baseline algorithm for the current multi-objective problem; however, this is not to comment on the relative merits of different algorithms which tend to vary according to specifics of problem and comparison metrics. Das and Suganthan [24] provide a survey of developments and applications in the field of DE. There are quite a few extensions of the DE algorithm into the multi-objective paradigm, as can be seen in [24], [25], among others. The current work bears broad similarity, algorithmically, to the prior developments of Iorio and Li [26], [27] who in turn have adapted some features of NSGA-II replacing the GA operations with corresponding DE steps.

B. The DIET Middleware

DIET [28] is an open-source middleware that enables a scalable execution of applications. Tasks are scheduled on distributed resources using a hierarchy of agents, as shown in Figure 1. DIET comprises several elements, including:

- **Client** application that uses the DIET infrastructure for remote problem solving.
- **Server Daemon** (***SeD***), which acts as a service provider exposing functionality through a standardized computational service interface. A single *SeD* can offer any number of computational services.
- **Agents**, deployed alone or in a hierarchy, facilitate service location and invocation interactions between clients and SEDs. Collectively, a hierarchy of agents provides high-level and scalable services such as scheduling and data management. The head of a hierarchy is termed as **Master Agent (MA)** whereas the others are **Local Agents (LA)**.

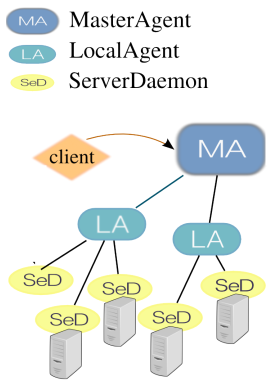

Fig. 1. An example of DIET Hierarchy

Applications are given a degree of control over the scheduling subsystem using plug-in schedulers (available in each agent) that use information gathered from resources via estimation functions (filled by each *SeD*). When a *SeD* receives a user request, by default it uses a pre-defined function to populate an estimation vector with system related information. A developer can create his own **performance estimation function** and include it into a *SeD* so that when the *SeD* receives a user request, the custom function is called to populate an estimation vector. These estimation vectors are used by agents to locate and invoke services required to execute a user application. Typically, a client request is made to a MA, which in turn broadcasts it to its agent hierarchy.

Another feature used in this work is DIET workload management capabilities. The DIET engine can handle workflow by assigning tasks to SeDs using one DIET service call. This assignment is made internally and dynamically by the MA, which receives requests from clients containing the description of a workflow. In this context, the MA determines how to schedule the workflow according to:

- Precedence constraints between tasks
- Scheduling policies/current plug-in schedulers
- Service performance properties
- Available resources on the infrastructure

This work uses the design of a new DIET plug-in scheduler to express information about servers' performance and power consumption, which is then taken into account when servers are provisioned to applications. Estimation vectors are used to determine the suitability of different SEDs while considering energy efficiency for executing the workflow and performance when executing the optimization engine service.

C. Related Work

Several approaches using multi-objective optimization to manage workload placement are present in the literature [29], [30]. Objectives refers to load balancing [31], load prediction [32] or platform reconfiguration [33], [34], among others. In [35], authors modelize the workflow scheduling problem as an evolutionary optimization problem with specification of genetic operators and simulations on a set of Real-World and random workflows. Fuzzy theory over a Pliant logic approach is used in [36] to improve energy efficiency in simulation-based experiments. Authors concluded with the need to find trade-offs between energy consumption and execution time for optimization. We used in our study a similar set of traces with a different interpretation of the task duration (cf Section VI).

Moreover, existing work [37] commonly assume that nodes from a homogeneous cluster consume the same amount of power, which is not always true in practice. Due to their different uses, nodes from a cluster can present different ranges of performance and power consumption. Causes of such differences include external environment factors, such as temperature and node location in a rack, aging of hardware components due to intensive use and leakage power that varies over time [38], [39]. We conclude from those studies that scheduling decisions based on performance and energy consumption values of the machines should be evaluated and dynamically adjusted using live monitoring.

From a resource management perspective, Grids and Clouds use meta schedulers to schedule jobs across multiple sites and local resource managers that control compute resources at a site level. Users commonly submit batch jobs to request resources over a period [40]. Cloud aggregators such as RightScale[1] provide application-specific Cloud management and load balancing. At an application level, distributed OS such as [41] offer programming models that allow OS services to scale to match demand. Most of these systems, however, neither take energy efficiency into account nor offer means for users to specify how they want to schedule their applications while exploring trade-offs between energy efficiency and performance [42].

III. NON-DOMINATED SORTING DIFFERENTIAL EVOLUTION II (NSDE-II)

Differential Evolution (DE) belongs to the broad class of evolutionary optimization techniques that developed as distinctive variants of classical Genetic Algorithms. This class of features has certains common features with Genetic Algorithm optimisation, namely, they work in parallel on a population of candidate solutions, are stochastic in nature, do not require the objective function to be analytic or even mathematically tractable, and are much less likely to get stuck in local optima as compared to gradient based methods. They differ from one another primarily in the manner in which candidate solutions in a new generation are synthesized from solutions in the current one, which effectively translates into their method of search of the total solution space for a global solution. The fitness of each candidate, as defined by one (or more) objective function(s), plays an important part in the evolution.

The core idea in DE is to superpose the difference between two randomly selected solution vectors (where the elements of a vector correspond to the values in dimensions of the solution space) on a third solution vector with each solution vector being a member of candidate population sets to obtain a new solution. Initially when (and if) the candidate solutions are spread across the solution space, the differences and hence the changes in solution vectors are large, and as the solutions converge to the global optimum, the changes get finer enabling attainment of the optimum faster. This is in contrast to classical GA where the changes on a solution vector are neutral to the level of evolution towards the global optimum.

This section presents the concept of differential evolution for a single objective and the key aspects to adapt it to Multi-Objective Differential Evolution.

1) Baseline Differential Evolution: Formally, if the dimensionality of the solution space is denoted as D and the number of candidate solutions is N, then the elements of the i^{th} vector of the solution $X_{i,G}$ at generation G may be denoted as

$$X_{i,G} = (X_{1,i,G}, X_{2,i,G}, X_{3,i,G}, ..., X_{D,i,G}) \quad for \quad all \quad i \in \mathbb{N} \tag{1}$$

The Differential Evolution (DE) process fundamentally generates new solutions from the current candidate set by adding the weighted difference between two randomly selected candidate solution vectors to a third to generate a mutant vector, and then creating a crossover between an existing vector and the mutant that is called the "trial" vector. The latter is allowed to replace the existing vector only if it is found

[1]Rightscale: http://www.rightscale.com/

to be more fit, the complexity of this fitness determination exercise depending entirely upon the nature of the problem under consideration. If $V_{i,G}$ represents the mutant vector, then according to the baseline DE process called DE/rand/1 [17], [18], [25]

$$V_{i,G} = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G}) \tag{2}$$

where $r1$,$r2$ and $r3$ are random integers less than N, different from each other and from i, and F usually lies between 0.5 and 1. There are many variations of this baseline process where two instead of one difference terms are sometimes considered, the best solution in a population is taken into account, etc.; descriptions of alternative schemes may be seen in [24], among others.

Crossover is performed between the mutant vector $V_{i,G}$ and the target vector $X_{i,G}$ to generate a trial vector $Z_{i,G}$ according to

$$z_{j,i,G} = \begin{cases} v_{j,i,G} & if \quad rand_j(0,1) \leq C_r \\ x_{j,i,G} & otherwise \end{cases} \tag{3}$$

where $z_{j,i,G}$ is the element j of the trial vector $Z_{i,G}$, $rand_j(0,1)$ denotes a random number between 0 and 1 applied to the element j, C_r is the crossover threshold usually set between 0.4 and 1. Eq. (3) simply states that the element j of the trial vector $Z_{i,G}$ is taken from the mutant vector if the corresponding random number generated with seed j is less than C_r, else the original value is left unchanged for that element.

At the final selection step the choice for candidate i in the next generation is made between $Z_{i,G}$ and $X_{i,G}$ on the basis of higher fitness by direct one-to-one comparison.

The present work generates the mutant vector according to the alternate scheme (proposed in [18] and also used by current authors in [22] where it is found to work better than other DE variants)

$$X_{i,G} = X_{r1,G} + R \times (X_{best,G} - X_{r1,G}) + F \times (X_{r2,G} - X_{r3,G}) \tag{4}$$

where R is set at 0.5 and F varies randomly between -2 and +2 across generations (and are same for all i within a generation). The crossover probability Cr in eq. 3 is set at 0.7.

2) Multi-Objective Differential Evolution NSDE-II: Compared to single-objective Differential Evolution (DE), the mechanisms for multi-objective DE are radically different. The basis for this difference follows from the altered conditions of selection that relate to this statement in the section above the "choice for candidate i in the next generation is made between $Z_{i,G}$ and $X_{i,G}$ on the basis of higher fitness by direct one-to-one comparison". That works for a single objective which tags a fitness value to both the solutions, enabling comparison. But if there is more than one objective, it is quite possible that one of them is more fit with respect to one objective, and the second for some other objective. And hence one cannot conclude which solution is more fit, upsetting the basic mechanism of single-objective DE.

This work has adopted the basic multiple-objective selection techniques of NSGA-II [14] while replacing the baseline GA operations to those of the DE variant outlined in Eqs. (3) and (4) for generation of a trial vector. Hence this is named as NSDE-II. In a problem with K objectives $FF_k, k \in 1...K$, a candidate solution vector X_p is said to dominate another solution X_q if

$$FF_k(X_p) \geqslant FF_k(X_q), k \in 1...K \tag{5}$$

and for at least one k, $FF_k(X_p) > FF_k(X_q)$; where $p, q \in 1...N$, N is population size; and in turn X_q is said to be dominated by X_p. This definition is used immediately below.

Now it is apparent that for a population of candidate solutions and with multiple objectives, there will be either one of three types of relations between any pair of candidate solutions. Either one dominates the other according to Eq. (5), or one is dominated by the other (i.e. converse to the first relation), or neither dominates or is dominated by the other, i.e. for some objectives one is better and for the balance objectives the other is the better.

This brings us to the NSDE-II selection process from one generation to the next.

Steps of Non-dominated selection:

1) A population of size N, taken for all parent vectors and all trial vectors thus forming a collective pool of size $2N$.
2) To every pool-member i allocate a number n_i and a vector S_i, where the former denotes the number of members that dominate it and the latter contains the identification index of all members that it dominates. This implies that S_i is a set whose size can vary from the null to at most $2N - 1$.
3) Place all members having $n_i = 0$ into a sub-pool called Front $F1$, which is an accumulation of the fittest members (i.e. those not dominated by any others). Thus the original pool is now depleted by the number of members shifted to $F1$.
4) Traversing all i put in F1, go over all the members j that are listed in S_i and reduce the corresponding value of n_j by 1. This implies that once a non-dominated member is shifted out of the pool, each remaining member of the depleted pool who was originally dominated by that removed member, is now dominated by one less member in this pool.
5) Now repeat steps 2-4, with the rider that the new set of $n_i = 0$ members (that emerge upon reducing the cardinality of domination by one) are put into the second front $F2$, and then $F3$, and so on.
At this point we have segregated the members of pool into a series of fronts with descending degree of non-dominance.
6) If the size of front $F1$ is less than N, select all members of $F1$ into the next generation.
7) Now if the size of front $F2$ is such that $\#(F1 + F2) < N$, then select all members of F2 also into the next generation (symbol # denotes cardinality of a set).
8) In this way move on to $F3$, $F4$ till one comes to some F_q where the size say s_q is larger than the number of unfilled spaces in gen-next, say u_q, i.e. $u_q = \left\{N - \sum_{m=1}^{q-1} \#F_m\right\}$, and $s_q > u_q$

9) Use the Crowding Distance Algorithm to select u_q out of s_q.

The core concept of the **Crowding Distance Algorithm** [14] is to select solutions that maximize diversity, i.e. if one solution is in a dense zone with many other solutions around, and another in a relatively sparse zone, then other aspects being equal, the solution from the sparse zone is selected. The algorithm quantifies the density of a point in terms of the distance between its two straddling neighbors in every dimension of the objective space, rather than of the parameter space.

IV. Problem Formulation

The aim of this work is to improve the energy efficiency of a set of machines while concurrently reducing completion time of a given set of jobs, through optimized workload placement. A server (computing node) is modeled with three resources: CPU, DISK and NETWORK and runs processes which consume these resources. Each process is to be assigned to a single machine, and cannot be moved from one machine to another.

A. Decision parameters

Any optimization problem will have design parameters whose best possible values from the viewpoint of the objectives are sought to be attained in the optimization process. The optimization task here is to map a given set of tasks in a certain sequence onto the available resources.

Suppose there are m number of resources and n tasks. Then, for any resource j, $\forall j \in [1...m]$, all possible permutations of subsets of all sizes of the set of tasks of size n, constitute the total solution space. If we call the size of this solution space as S_j, then

$$S_j = \sum_{k=0}^{n} P(n,k) \qquad where \qquad P(n,k) = \frac{n!}{(n-k)!} \tag{6}$$

Since S_j is independent of j, we may write it simply as S. It then follows that the size of the total solution space is m^S.

The following information is assumed to be known for each server s at any time:

f_s	Number of FLoating-point Operations Per Second (FLOPS)
dw_s	Disk Writing rate
dr_s	Disk Reading rate
net_s	Available Network bandwidth
c_s	Average power consumption
nf_i	Number of FLOPS to perform the task i.
nbw_i	Number of bytes written on disk by the task i.
nbr_i	Number of bytes read on disk by the task i.
$nnet_i$	Number of bytes exchanged by the task over the network by the task i.

The knowledge of these variables enables the scheduler to consider the energy efficiency related to the completion of tasks.

B. Objective functions

We have two objective functions:

1) Minimize Makespan (i.e. time taken for completion of all tasks in the workflow)
If T_i is the completion time of the task i, then

$$T_i = \frac{nf_i}{f_s} + \frac{nbw_i}{dw_s} + \frac{ndr_i}{dr_s} + \frac{nnet_i}{net_s} \tag{7}$$

The completion time of the workflow is expressed as

$$T_{mn} = \sum_{1}^{m}\sum_{1}^{n} T_i \tag{8}$$

2) Minimize Energy consumption (i.e. total energy consumed in a workflow)
If $C_s(T_i)$ is the energy consumption of the task i per unit time when running on server s, then energy consumption required for the workflow is expressed as

$$W_s i = \sum_{1}^{m} C_s(T_i) \times T_i \tag{9}$$

and the total energy consumed is

$$W_{mn} = \sum_{1}^{n}\sum_{1}^{m} C_s(T_i) \times T_i \tag{10}$$

In most cases, faster machines (low T_i) will have higher energy consumption (high C_s), implying that objectives T_{mn} and W_{mn} are contradictory — forming the basis of multi-objective optimization.

V. Framework for Workload Placement

To cope with real conditions such as the increasing scale of modern data centers, as well as the workload dynamics and application characteristics that are specific to the Cloud Computing paradigm, DIET allows users to study large-scale scenarios that involve thousands of nodes, each executing a specific workload that evolves during the computation.

The aim of the current framework is twofold: (i) to relieve researchers of the burden of dealing with deployment, resource selection and workload fluctuations when they evaluate new optimization engines and (ii) to offer the possibility to compare them. This work adds to the infrastructure defined in prior works from authors in [7]. To perform placement decisions, users encapsulate their optimisation engine in a program, and express the workflow along with the precedence between tasks. The program typically leverages DIET API that allows end users to create and execute remote services[2]. The MasterAgent keeps a description of the physical resources, dynamically updated by the nodes hosting the services. Finally, the workload execution is orchestrated by the DIET workflow engine that internally relies on a customizable scheduler (cf. Section II) to assign the resources during the entire execution. We chose to base our framework on DIET since (i) the latters relevance in terms of performance and validity has already been demonstrated [43] and (ii) because it has been recently

[2]http://graal.ens-lyon.fr/diet/UsersManualDIET2.9/

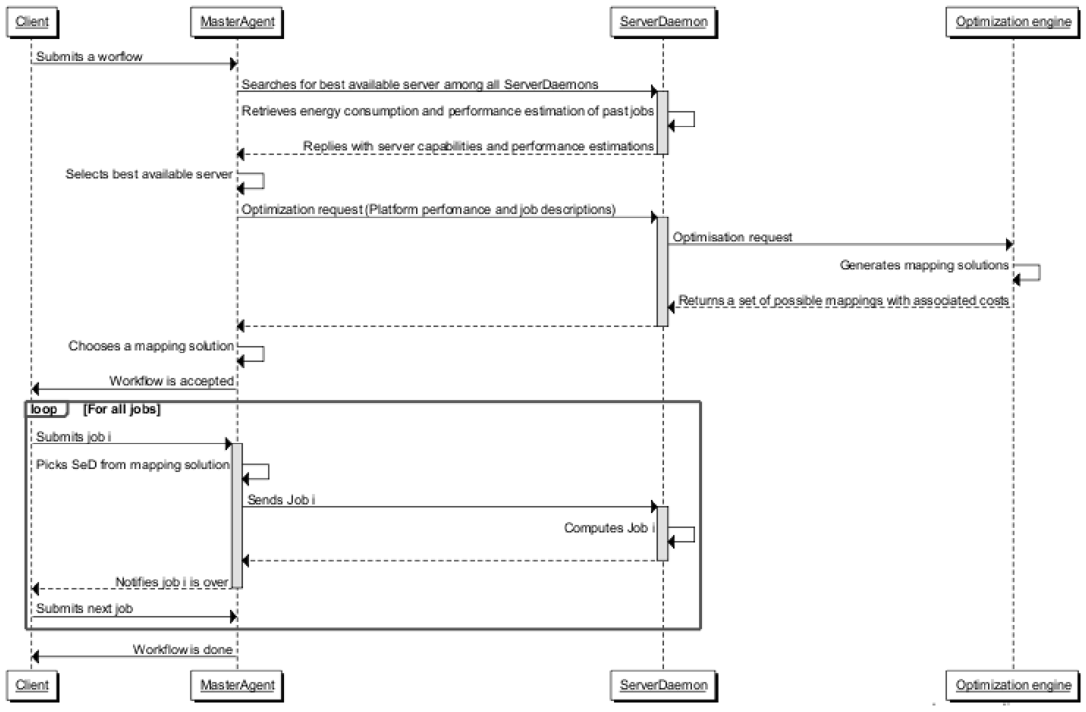
Client
MasterAgent
ServerDaemon
Optimization engine
Submits a worflow
Searches for best available server among all ServerDaemons
Retrieves energy consumption and performance estimation of past jobs
Replies with server capabilities and performance estimations
Selects best available server
Optimization request (Platform perfomance and job descriptions)
Optimisation request
Generates mapping solutions
Returns a set of possible mappings with associated costs
Chooses a mapping solution
Workflow is accepted
loop
[For all jobs]
Submits job i
Picks SeD from mapping solution
Sends Job i
Computes Job i
Notifies job i is over
Submits next job
Workflow is done

Fig. 2. Optimisation and workflow execution equence

extended to integrate energy-efficient decisision capabilities [7].

The workflow execution is performed in 3 phases (Figure 2): (i) service discovery, (ii) computation of mapping solutions and (iii) workload placement. The service discovery phase corresponds to the search of an optimization engine within the infrastructure by a given client. As multiple engines can be instantiated on the platform, the user can submit its workload to different engines and compare the cost of generated solutions. The computation of mapping solutions is performed by at least one server (multiple servers can be put in cooperation using the same service, based on the engine requirements) with a platform performance description provided by the Master Agent. This description is either based on historical data (past computations) or user-defined benchmarks. Finally, the workload placement is performed and results are returned to the client based on the platform available metrics and monitoring resolution. Mapping solutions are defined as a collection of JSON objects. Each solution contains the mapping between a *SeD* and a task and an associated cost in terms of workflow completion time and energy consumption.

Two kinds of experiments have been performed to validate this approach. The objective of the first one is to evaluate the computation phase of the engine (i.e., the step where the optimization engine generates a spectrum of solutions) while the second is a comparison of algorithms to evaluate the concrete gain of NSDE-2 compared to an online placement of workload.

VI. EXPERIMENTAL SETUP AND VALIDATION

Experiments used resources from GRID'5000 [44], a testbed designed to support experiment-driven research in parallel and distributed systems. Located in France, GRID'5000 comprises 29 heterogeneous clusters, with 1,100 nodes, 7,400 CPU cores with various generations of technology spanning 10 physical sites interconnected by a dedicated 10 Gbps backbone network. By providing bare-metal resource deployment, GRID'5000 enables users to experiment on all layers of the software stack of distributed infrastructures, including high-performance computing, grids, peer-to-peer, and Cloud computing architectures.

The power measurement in the studied clusters is performed with an energy-sensing infrastructure composed of external wattmeters produced by the SME Omegawatt. This energy-sensing infrastructure, also used in previous work [7], [45], collects at every second the power consumption in averaged watts of each monitored node [46]. A node's consumption is determined by averaging past consumption over more than 6,000 measurements, whereas its performance is given by the number of FLOPS achieved when using all CPU cores to execute benchmarks are using ATLAS[3], HPL[4] and Open MPI[5].

We have used real-world trace files of an international company called Prezi Inc [6], who offers a presentation editing service, which is available on multiple platforms, therefore they have to convert some of their created media files to other formats before they can display them on all devices. Their conversion processes are carried out on virtual machines: at peak times, they need to launch more instances of these VMs, but over the weekend they can stop most of them. They published log files on their website containing workload

[3]Automatically Tuned Linear Algebra Software: http://sourceforge.net/projects/math-atlas/

[4]Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers: http://www.netlib.org/benchmark/hpl/

[5]High Performance Message Passing Library: http://www.open-mpi.org/

[6]http://prezi.com/scale/

traces for two weeks of utilization, which serves as a basis for algorithmic experimentations. They operate three queues in their Cloud system for the jobs participating in the conversion processes:

- export: contains jobs which result in downloadable zipped Prezi files.
- url: these jobs download an image from a URL and insert them into a Prezi file.
- general: all other conversion jobs (audio, video, pdf, ppt, etc.).

The lines of the published workload traces have the following format:

2012-12-14 21:35:12 237 general 9.134963

This means that at the given time, a job enters the general queue with the id 237, and the job will take 9.134963 seconds to run. The available trace files contain more than 2000000 lines, and their submitted (and processed) jobs highly varies over the 14 days.

To represent the job heterogeneity and their hardware requirements, we created a generic multi-thread program in charge of interpreting and executing requests based on a log trace description. Each task is represented by an execution of a bounded number of operations.

An operation is based on the completion of three functions, simultaneously executed by three different threads:

- A CPU-intensive operation consisting in the multiplication of two randomly filled matrices of size 1000x1000 (*cpu*).
- A disk-intensive operation consisting in the writing and reading on disk of a 20MB file (*disk*).
- A network-intensive operation consisting in downloading a 5MB file from a remote server (*net*).

In the context of this experiment, each of the three threads is in charge of the sequential execution of n functions of the same type, n being the weight of each function. Each queue has a weighted sum of functions. We consider the following mapping for each type of job:

export	$2 \times cpu + 1 \times disk + 1 \times net$
url	$1 \times cpu + 2 \times disk + 3 \times net$
general	$3 \times cpu + 1 \times disk + 1 \times net$

As an example, a job with the id 237 from the general queue will be completed after the execution of 10 general operations.

We deploy the DIET middleware on 113 physical nodes as follows: 111 dedicated nodes for *SeD*'s, 1 dedicated node for the Master Agent and 1 dedicated node for the Client. The machines are picked among six different clusters as presented in Table I.

A. NSDE-2 engine

NSDE-2 being an evolutionary multi-objective algorithm works on a population of candidate solutions which improve on all objectives across generations. The solution at any generation is presented in the form of a Pareto front which represents the position of each candidate in the multi-objective reference frame. Here we work on a population of size 200. Figure 3 shows the evolution of the Pareto front from an initial stage of 100 generations up to 10000 generations, with minimization of energy consumption and makespan as the objectives. Each dot represents a candidate solution. At any selected generation, at one end of the Pareto front we have the best energy solution, and at the other end, the best makespan solution.

We can observe that the quality of solutions improves as the number of generations increases. The computation time increases linearly as the number of generations increase. We choose to retrieve the solution at 3000 generations, after which the improvement is solution becomes less significant.

It may be noted that if jobs are submitted on the cloud for execution after prior reservation, then it can be valuable to drive the NSDE-2 to its full potential to obtain the best optimal solution placement.

TABLE I. EXPERIMENTAL INFRASTRUCTURE.

Cluster	Nodes	CPU	Memory	Role
Orion	4	2x6 cores @2.30Ghz	32GB	*SeD*
Sagittaire	38	2x1 core @2.40Ghz	2GB	*SeD*
Taurus	10	2x6 cores @2.30Ghz	32GB	*SeD*
Stremi	38	2x12 cores @1.7Ghz	48GB	*SeD*
Graphite	4	2x6 cores @2.00Ghz	64GB	*SeD*
Parasilo	17	2x6 cores @2.40Ghz	128GB	*SeD*
Parasilo	1	2x6 cores @2.40Ghz	128GB	MA
Parasilo	1	2x6cores @2.40Ghz	128GB	Client

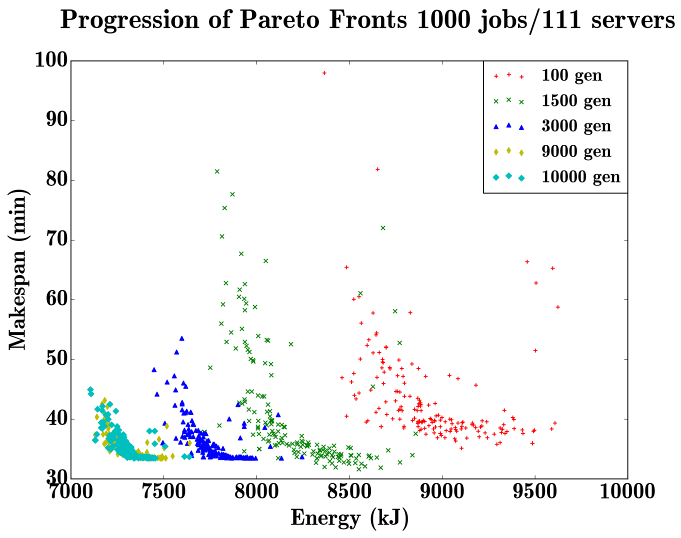

Fig. 3. Evolution of the solutions as the number of generations increases

B. Scalability and Reactivity of NSDE-II

Figure 4 shows improvements in NSDE-2 execution time with increasing parallelization when the solutions are per-

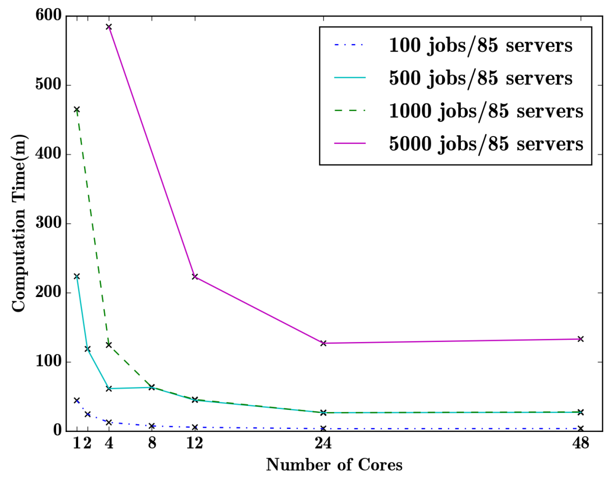

Fig. 4. NSDE-2 execution time for generating mapping solutions related to 4 datasets and 111 servers

formed on a Stremi node (cf. Table I).

C. Workload placement

This evaluation aims to compare the distribution of tasks among nodes on GRID'5000 considering three different policies, namely NSDE-2 Best Energy, NSDE-2 Best Performance and FIRST FIT. NSDE-2 Best Energy and NSDE-2 Best Performance correspond, respectively, to the smallest energy consumption and the smallest makespan. These solutions establish the bounds of the Pareto Front. The FIRST FIT policy consists in the selection of the first available server in an ordered list accroding to the $GreenPerf$ metric as a non-weighted average ratio between performance and energy consumption for the said type of task.

In each scenario, we consider the first entries of the trace file. For any of considered cases there exists a proper balance between short and long tasks within the dataset. A server is restricted to the execution of, at most, one task at a given time.

Considering that the scheduler does not have specific information on the nodes and does not make assumptions about the hardware, the dynamic information is gathered as a sample of each tasks is computed by the servers prior to the experiment. Figures 5, 6, 7 and 8 show the results of this experiment. The x-axis presents the different algorithms used to execute the workflow; the y-left-axis shows the total energy consumption of the solution and the y-right-axis shows the makespan value.

We observe an influence of the ratio jobs/servers on the global results. The larger the dataset (specifically in terms of large tasks), the worst FIRST FIT performs as it prevents the packing of tasks on the most energy-efficient nodes, resulting in more uses of least energy-efficient nodes, thus in higher energy consumption. On small dataset, this effect is hidden by the fact that fast nodes will compute more tasks in parallel.

Tables II,III,IV show actual values obtained in terms of energy and time for the three allocation policies, for the 4

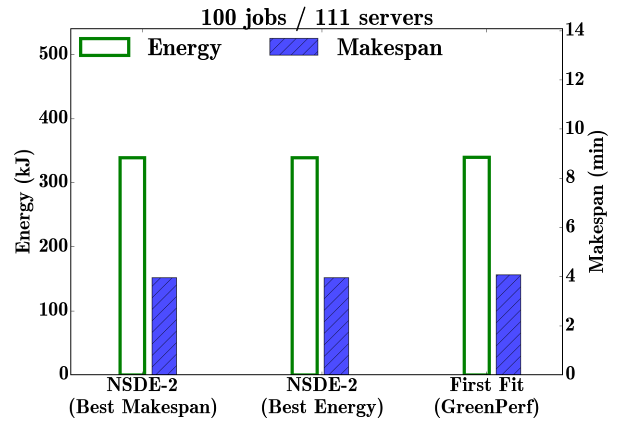

Fig. 5. Energy and Makespan comparison for 100 jobs and 111 servers

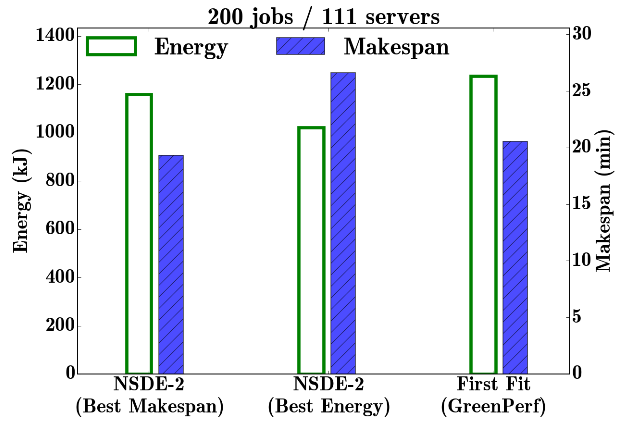

Fig. 6. Energy and Makespan comparison for 200 jobs and 111 servers

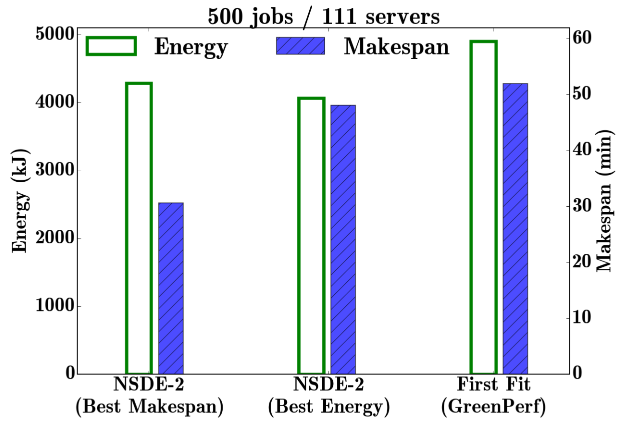

Fig. 7. Energy and Makespan comparison for 500 jobs and 111 servers

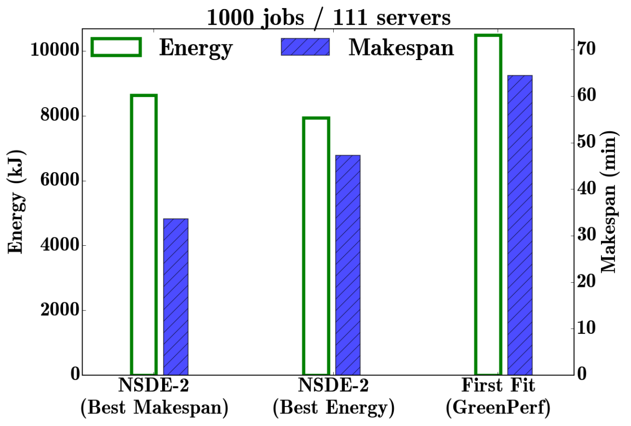

Fig. 8. Energy and Makespan comparison for 1000 jobs and 111 servers

TABLE II. ENERGY CONSUMPTION COMPARISON

Cases	No of Jobs	NSDE-2 Best Makespan (kJ)	NSDE-2 Best Energy (kJ)	First Fit *GreenPerf* (kJ)
1	100	338.9	338.9	339.8
2	200	1158.1	1020.3	1233.3
3	500	4287.1	4067.7	4901.7
4	1000	8632.5	7943.5	10482

illustrated cases with 111 servers. It may be seen that NSDE-2 improves for any of the scenarios (except in the smallest case (Figure 5), and for the larger cases improves makespan as well. The user may choose to select an intermediate solution on the Pareto front that improves both energy and makespan, trading-off between the two objectives. It is worth noting that when the NSDE-2 solution was run up to 10000 generations, it provided a 30% saving in energy with a 50% reduction in makespan. Considering the computation time of the Pareto Front, this can be of value in cases of jobs submitted by prior reservation.

TABLE III. MAKESPAN COMPARISON

Cases	No of Jobs	NSDE-2 Best Makespan (m)	NSDE-2 Best Energy (m)	First Fit *GreenPerf* (m)
1	100	3.94	3.94	4.07
2	200	19.33	26.6	20.56
3	500	30.63	48.0	51.89
4	1000	33.64	47.29	64.47

TABLE IV. COMPARATIVE IMPROVEMENTS USING NSDE-2 IN MAKESPAN AND ENERGY

Cases	No of jobs	NSDE-2 Computation time of solutions (m)	Makespan NSDE-2 Performance (%)	Makespan NSDE-2 Energy (%)	Energy NSDE-2 Performance (%)	Energy NSDE-2 Energy (%)
1	100	3.46	-82	-82	0	0
2	200	6.0	-23	-59	6.1	17.3
3	500	13.63	15	-19	12.5	17.1
4	1000	26.5	7	-14	17.6	24.3

VII. Conclusion and perspectives

In this work, we report on design, implementation and evaluation of an energy-efficient resource management system that build upon DIET, an open source middleware and NSDE-II, a an Evolutionary Multi-Objective Optimization engine. Our implementation supports an IaaS Cloud and currently provides placement of workflows, considering non-divisble tasks with precedences constraints. Real-life experiment of our approach on the GRID'5000 testbed demonstrates the effectiveness of our approach in a dynamic environnment. Results shows that our method can provide providers and decision makers an aid to make their decision when conflicting objectives are present or when in search for realistic tradeoff for a given problem.

As future works, it would be valuable to add multi-core integration. This is important as we can expect a significant reduction in energy consumption when multiple tasks are loaded in parallel on a machine.

VIII. Acknowledgments

This research has been supported in part by CEFIPRA (Indo-French Center for promotion of Advanced Research) through its Raman-Charpak Fellowship program. Some experiments presented in this paper were carried out using the Grid5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] I. Foster and C. Kesselman, "Computational grids," in *Vector and Parallel ProcessingVECPAR 2000*. Springer, 2001, pp. 3–37.

[2] J. Dongarra *et al.*, "The International Exascale Software Project roadmap," *The International Journal of High Performance Computing Applications*, vol. 25, no. 1, pp. 3–60, Feb. 2011.

[3] A. Hawkins, "Unused servers survey results analysis," *The Green Grid*, 2010.

[4] A.-C. Orgerie, M. D. d. Assuncao, and L. Lefèvre, "A survey on techniques for improving the energy efficiency of large-scale distributed systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 47, 2014.

[5] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Trans. Parallel Distrib. Syst*, vol. 24, no. 7, pp. 1366–1379, 2013.

[6] A.-C. Orgerie and L. Lefèvre, "When Clouds become Green: the Green Open Cloud Architecture," *Parallel Computing*, vol. 19, pp. 228 – 237, 2010. [Online]. Available: http://hal.inria.fr/ensl-00484321

[7] D. Balouek-Thomert, E. Caron, and L. Lefevre, "Energy-aware server provisioning by introducing middleware-level dynamic green scheduling," in *Workshop HPPAC'15. High-Performance, Power-Aware Computing*. Hyderabad, India: In conjunction with IPDPS 2015, May 2015.

[8] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms." in *Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985*, 1985, pp. 93–100.

[9] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. Ieee, 1994, pp. 82–87.

[10] J. Knowles and D. Corne, "The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 1. IEEE, 1999.

[11] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *evolutionary computation, IEEE transactions on*, vol. 3, no. 4, pp. 257–271, 1999.

[12] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[13] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improved the performance of the strength pareto evolutionary algorithm," Technical Report 103, Computer Engineering and Communication Networks Lac (TIK), Swiss Federal institute of Technology (ETH) Zurich, Tech. Rep., 2001.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[15] A. K. Bhattacharya and D. Sambasivam, "Optimization of oscillation parameters in continuous casting process of steel manufacturing: Genetic algorithms versus differential evolution," in *Evolutionary Computation*. InTech, DOI: 10.5772/9616, 2009.

[16] J. H. Holland, "Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence." 1975.

[17] R. Storn and K. Price, *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. ICSI Berkeley, 1995, vol. 3.

[18] ——, "Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[19] J. Kennedy, "R, eberhart," *Particle swarm optimization*, vol. 1, 1995.

[20] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 1, pp. 29–41, 1996.

[21] M. Dorigo and L. Gambardella, "Ant-q: A reinforcement learning approach to the traveling salesman problem," in *Proceedings of ML-95, Twelfth Intern. Conf. on Machine Learning*, 2014, pp. 252–260.

[22] A. K. Bhattacharya, D. Aditya, and D. Sambasivam, "Estimation of operating blast furnace reactor invisible interior surface using differential evolution," *Applied Soft Computing*, vol. 13, no. 5, pp. 2767–2789, 2013.

[23] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2, June 2004, pp. 1980–1987 Vol.2.

[24] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 4–31, 2011.

[25] C. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media, 2007.

[26] A. W. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in *AI 2004: Advances in artificial intelligence*. Springer, 2005, pp. 861–872.

[27] ——, "Incorporating directional information within a differential evolution algorithm for multi-objective optimization," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2006, pp. 691–698.

[28] E. Caron and F. Desprez, "DIET: A scalable toolbox to build network enabled servers on the grid," *International Journal of High Performance Computing Applications*, vol. 20, no. 3, pp. 335–352, 2006.

[29] A. Talukder, M. Kirley, and R. Buyya, "Multiobjective differential evolution for scheduling workflow applications on global grids," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 13, pp. 1742–1756, 2009.

[30] J.-T. Tsai, J.-C. Fang, and J.-H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," *Computers & Operations Research*, vol. 40, no. 12, pp. 3045–3055, 2013.

[31] A. Abdulmohson, S. Pelluri, and R. Sirandas, "Energy efficient load balancing of virtual machines in cloud environments," 2015.

[32] R. Alkharboush, R. E. De Grande, and A. Boukerche, "A genetic algorithm approach for adjusting time series based load prediction," in *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*. IEEE, 2015, pp. 292–298.

[33] F. Legillon, N. Melab, D. Renard, and E.-G. Talbi, "A multi-objective evolutionary algorithm for cloud platform reconfiguration," in *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*. IEEE, 2015, pp. 286–291.

[34] S. Frey, F. Fittkau, and W. Hasselbring, "Search-based genetic optimization for deployment and reconfiguration of software in the cloud," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 512–521.

[35] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud."

[36] A. Benyi, J. D. Dombi, and A. Kertesz, "Energy-aware vm scheduling in iaas clouds using pliant logic," in *proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER14), Barcelona, Spain*, 2014.

[37] K. H. Kim, R. Buyya, and J. Kim, "Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters," in *CCGRID*. IEEE Computer Society, 2007, pp. 541–548.

[38] G. Varsamopoulos, A. Banerjee, and S. K. S. Gupta, "Energy efficiency of thermal-aware job scheduling algorithms under various cooling models," in *Contemporary Computing - Second International Conference, IC3 2009, Noida, India, August 17-19, 2009. Proceedings*, ser. Communications in Computer and Information Science, vol. 40. Springer, 2009, pp. 568–580.

[39] M. E. M. Diouri *et al.*, "Your cluster is not power homogeneous: Take care when designing green schedulers!" in *IGCC-4th IEEE International Green Computing Conference*, 2013.

[40] N. Capit and al., "A batch scheduler with high level components," in *Cluster computing and Grid 2005 (CCGrid05)*, 2005.

[41] D. Wentzlaff *et al.*, "An operating system for multicore and clouds: Mechanisms and implementation," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 3–14.

[42] C.-Y. Tu, W.-C. Kuo, W.-H. Teng, Y.-T. Wang, and S. Shiau, "A power-aware cloud architecture with smart metering," in *Proc. Second International Workshop on Green Computing (2nd GreenCom'10), 2010 International Conference on Parallel Processing Workshops (39th ICPPW'10) CD-ROM*. San Diego, CA: CPS/IEEE Computer Society, Sep. 2010, pp. 497–503.

[43] E. Caron, B. Depardon, and F. Desprez, "Deployment of a hierarchical middleware," in *Euro-Par 2010*, LNCS, Ed., vol. 6271 Part I. Ischia - Naples, Italy: Institute for High Performance Computing and Networking of the Italian National Research Council, August 31 to September 3 2010, pp. 343–354.

[44] F. Cappello, E. Caron, M. Dayde, F. Desprez, E. Jeannot, Y. Jegou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, and O. Richard, "Grid'5000: a large scale, reconfigurable, controlable and monitorable Grid platform," in *SC'05: Proc. The 6th IEEE/ACM International Workshop on Grid Computing Grid'2005*. Seattle, USA: IEEE/ACM, Nov. 2005, pp. 99–106.

[45] M. D. De Assuncao, A.-C. Orgerie, and L. Lefèvre, "An analysis of power consumption logs from a monitored grid site," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*. IEEE, 2010, pp. 61–68.

[46] M. D. de Assuncao, J.-P. Gelas, L. Lefèvre, and A.-C. Orgerie, "The green grid'5000: Instrumenting and using a grid with energy sensors," in *Remote Instrumentation for eScience and Related Aspects*. Springer, 2012, pp. 25–42.