Parallel Differential Evolution approach for Cloud workflow placements under simultaneous optimization of multiple objectives

Daniel Balouek-Thomert, Arya K. Bhattacharya, Eddy Caron, Karunakar Gadireddy, Laurent Lefèvre

Parallel Differential Evolution approach for Cloud workflow placements under simultaneous optimization of multiple objectives

Daniel Balouek-Thomert*[†], Arya K. Bhattacharya[‡#], Eddy Caron[†], Karunakar Gadireddy[‡], Laurent Lefevre[†]

* 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

[#] Senior Member, IEEE

***Abstract*— The recent rapid expansion of Cloud computing facilities triggers an attendant challenge to facility providers and users for methods for optimal placement of workflows on distributed resources, under the often-contradictory impulses of minimizing makespan, energy consumption, and other metrics. Evolutionary Optimization techniques that from theoretical principles are guaranteed to provide globally optimum solutions, are among the most powerful tools to achieve such optimal placements. Multi-Objective Evolutionary algorithms by design work upon contradictory objectives, gradually evolving across generations towards a converged Pareto front representing optimal decision variables – in this case the mapping of tasks to resources on clusters. However the computation time taken by such algorithms for convergence makes them prohibitive for real time placements because of the adverse impact on makespan. This work describes parallelization, on the same cluster, of a Multi-Objective Differential Evolution method (NSDE-2) for optimization of workflow placement, and the attendant speedups that bring the implicit accuracy of the method into the realm of practical utility. Experimental validation is performed on a real-life testbed using diverse Cloud traces. The solutions under different scheduling policies demonstrate significant reduction in energy consumption with some improvement in makespan.**

Keywords: Parallel computing; workflow placement; cloud computing; energy; makespan; middleware; Differential Evolution; multi-objective evolutionary algorithm; Pareto front convergence.

I. INTRODUCTION

Many of the IT and Analytics services that organizations utilize currently 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 computing resources in their datacentres [2]. This situation is partially caused by an overprovisioning to ensure service delivery at peak hours, leading to underutilized resources at other times [3]. Efficient allocation of tasks to resources can improve consolidation on a minimum number of nodes, while transiting remaining unused nodes to low-power modes or shutdown [4], [5]. The implementation of the task-to-resource allocation policy consists in picking in real time at Cloud provider's end the best combination of resources, in order to fit the customer's needs at lowest cost, risk and energy consumption. Server allocation policies usually involve two actors: the *Cloud provider* who defines placement policies according to available resources while the *customer* submits sets of tasks to be executed. Such policies must benefit both the provider and customer in terms of the above metrics.

In previous work [6], 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, considering the contradictory nature of these objectives, *GreenPerf* could not fully explore the large domain space of possible 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 solutions representing minimum energy at one end of the front and minimum makespan (completion time) at the other.

The field of multi-objective optimization, particularly techniques using evolutionary algorithms, has advanced significantly since the first attempt [7] using Genetic Algorithms, and is widely used today in numerous applications. Among the most noteworthy developments rank the SPEA2 algorithm by Zitzler et al [8] and NSGA-II algorithm by Deb et al [9]. 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.

Evolutionary Algorithms that work concurrently on a population of candidate solutions are naturally amenable to parallelization and consequent speedup, because a significant percent of the computations operate on individual candidates independent of the others. There are two broad paradigms for parallelization, the "master-slave" model [10] and the "island" model [11]. Talbi et al [12] provides a comparative analysis of various approaches towards implementation of parallelism for Multi-Objective Evolutionary Optimization.

This work focuses on workflow applications that consist of multiple components (tasks) related by precedence constraints that usually follow the data flow between them. 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 makespan as an output. This output is to be placed and executed on the infrastructure using the DIET Middleware. In this framework the time spent on NSDE-2 optimization contributes to the makespan, hence this work addresses speedup of NSDE-2 through parallelization. The current version uses only energy and makespan as the objectives for concurrent minimization, the intention is to gradually integrate more independent objectives into the optimization process.

This paper introduces several contributions: (i) an evolutionary approach to workflow placement (ii) a choice of solutions to the user based on his priorities, ranging from best-energy to best-makespan, and intermediates, (iii) an experimental protocol using a real life testbed and (iv) parallel launching of evolutionary optimization process on the same Cloud infrastructure targeted for workflow placement.

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. In Section VI we evaluate the quality of workflow placement using our approach. Finally we draw Conclusions and discuss further lines of development.

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 using multi-objective evolutionary optimization.

A. Differential Evolution

The developments in Multi-Objective Evolutionary Algorithms referred in Section I have been along the track of Genetic Algorithm (GA) [13], the baseline Evolutionary optimization approach, applied to the direct multi-objective paradigm. At the basic algorithm level, Differential Evolution (DE) was formulated as an alternate approach to GA by Storn and Price [14]. The present authors have applied both GA and DE in a few complex industrial processes [15-17]; the latter work also provides a comparison in computational efficiency for that industrial process between GA and DE demonstrating that DE comes out favourably. Due to these developments the authors decided to use their version of DE as the baseline algorithm for the current multi-objective problem.

Evolutionary Algorithms have also been successfully parallelized on Cloud frameworks. Lee et al [18] implemented a parallel GA-PSO method for inferring gene networks in a Cloud computing environment using the Hadoop MapReduce programming model. Tang et al [19] parallelized the DE algorithm using a resilient distributed datasets model, and compared consequent performance improvements relative to MapReduce on a wide range of benchmark problems. The above examples represent parallelization of single-objective evolutionary algorithms on Cloud clusters to solve specific optimization problems, and not scheduling of actual workflows based on multiple objectives.

B. The DIET Middleware

DIET [20] 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)**.

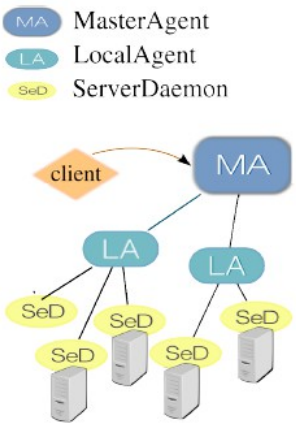


Fig. 1. An example of DIET Hierarchy

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 [21], [22]. Objectives refer to load balancing [23], load prediction or platform reconfiguration [24], among others. A Pliant logic approach is used in [25] to improve energy efficiency in simulation based experiments. The authors conclude with the need to find trade-offs between energy consumption and execution time for optimization. Although most of the above works deal with workflow scheduling on Clouds using Multi-Objective Evolutionary algorithms, they have not explored the parallelism potential of the Cloud

infrastructure in the scheduling process itself. One of the first developments in that direction is seen in [26], where a Genetic Algorithm is used for optimization and Dynamic Voltage Scaling to minimize energy consumption. A comprehensive review of the state of the field is presented in [27]; work on parallelism of Differential Evolution algorithms in this context is yet to be reported.

Moreover, existing work [28] commonly assume that nodes from a homogeneous cluster are identical in power consumption and performance, which is not always true in practice. Causes of variation include external environmental factors, such as temperature and node location in a rack, aging of components due to use and leakage power that varies over time [29]. We conclude 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 computational resources at a site level. Users commonly submit batch jobs to request resources over a period [30]. Cloud aggregators such as *RightScale* provide application-specific Cloud management and load balancing. At an application level, distributed OS such as [31] 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 [32].

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 (GA). DE was selected as the evolutionary method of choice on the basis of the authors' prior studies on the relative efficiency and merits of this against GA, as reported in [17].

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

A. 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

$$X_{i,G} = (x_{1,i,G}, x_{2,i,G}, x_{3,i,G,}, \ldots\ldots\ldots, x_{D,i,G})$$

for all i ∈ N … (1)

The 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 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 [14]

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

… (2)

where r_1, r_2 and r_3 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 [33], 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

$$z_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if } \text{rand}_j(0,1) \le Cr \\ x_{j,i,G} & \text{otherwise} \end{cases}$$

… (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 & 1 applied to the element j, Cr is the crossover threshold usually set between 0.4 and 1. 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 [14] and also used by current authors in [15-17] where it is found to work better than other DE variants)

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

(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.

B. Multi-Objective Differential Evolution NSDE-II

Compared to single-objective DE discussed in Sec. 3A, the mechanisms of selection to a new generation in multi-objective DE are radically different. The basis for this difference lies in the fact that one cannot uniquely order the candidate solutions based on their "fitness" when the number of axes for ordering is more than one. Thus when comparing two candidates $Z_{i,G}$ and $X_{i,G}$, the former may be better for the first objective and the reverse for the second. This problem on transiting from single- to multiple-objectives is equally relevant for any evolutionary algorithm and not for DE alone.

This work has adopted the basic multiple-objective handling techniques of NSGA-II [9] while replacing the baseline GA operations to those of the DE variant outlined in (3) and (4) for generation of a trial vector. Hence this is named as NSDE-II. It may be noted that in DE "elitism" and hence selection of the better performing solutions executes a more integral and critical role in the evolutionary process compared to typical GA where elitism is not mandatory.

In a problem with K objectives FF_k, k 1…K, a candidate solution vector X_p is said to dominate another solution X_q if

$$FF_k(X_p) \geq FF_k(X_q), \forall k \in \{1,...,K\} \quad \ldots (5)$$

$$FF_k(X_p) > FF_k(X_q)$$

and for at least one k, ; where p, q {1, …, N}, N is population size; and in turn X_q is said to be dominated by X_p.

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 (5), or one is dominated by the other, or neither dominates or is dominated by the other.

The NSDE-II algorithm exploits these three basic types of relationships to select candidates from the population pool into the next generation, by stratifying the pool into fronts based on descending degree of non-dominance. At a secondary level it also exploits the degree of diversity among solutions, giving preference to candidates with higher diversity, using the Crowding Distance Algorithm [9].

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 jobs in a certain sequence onto the available resources.

Suppose there are m number of resources and n tasks. Then, for any resource j, j [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 Sj , then

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

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, and task i, at any time:

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

The knowledge of these variables enables the scheduler to extract the energy consumption and makespan related to the completion of tasks over the available resources.

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_{is} is the completion time of the task i launched on server s, then

$$T_{is} = \frac{nf_i}{f_s} + \frac{nbw_i}{dw_s} + \frac{nbr_i}{dr_s} + \frac{nnet_i}{net_s}$$

… (7a)

If the task *i* is launched on a server other than *s*, then

$$T_{is} = 0$$

… (7b)

The completion time of the workflow is expressed as

$$T_{mn} = \sum_{s=1}^{m} \sum_{i=1}^{n} T_{is}$$

… (8a)

Eq. (8a) is the total time taken to execute all tasks on all servers. However, by definition, *makespan* is the time when the last server completes its task in the workflow. Hence, (8a) is modified to yield:

$$\text{Makespan } T_{mn} = \underset{s \in (1,..,m)}{Max} \sum_{i=1}^{n} T_{is}$$

… (8b)

2) Minimize Energy consumption (i.e. total energy consumed in a workflow)

If C_{is} is the energy consumption of the task *i* per unit time running on server s, then energy consumption on server *s* required for the workflow may be expressed as

$$W_s = \sum_{i=1}^{n} C_{is} T_{is}$$

… (9)

and the *total energy consumed* in the workflow is

$$W_{mn} = \sum_{s=1}^{m} W_s = \sum_{s=1}^{m} \sum_{i=1}^{n} C_{is} T_{is}$$

… (10)

In most cases, faster machines (low T_{is}) will have higher energy consumption (high C_{is}), implying that objectives T_{mn} and W_{mn} are contradictory – forming the basis for 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. 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[1]. The Master Agent 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 latter's relevance in terms of performance and validity has already been demonstrated [34] and (ii) because it has been recently extended to integrate energy-efficient decision capabilities [6].

The workflow execution is performed in 3 phases: (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 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

1 http://graal.ens-lyon.fr/diet/UsersManualDIET2.9/

concrete gain of NSDE-2 compared to an online placement of workload.

VI. EVALUATION OF WORKFLOW PLACEMENT

In this section we first briefly describe the evaluation testbed. Then we look at the approach and consequences of parallelization of NSDE-2 algorithm. Finally we evaluate the quality of workflow placement using this multi-objective evolutionary optimization approach compared to extant methods.

A. Evaluation Testbed

Experiments used resources from GRID'5000 [35], 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.

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 [6], [36], collects at every second the power consumption in averaged watts of each monitored node [37]. 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 a single CPU cores to execute benchmarks.

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. Detailed description of trace files and their usage in experiments are provided in a related work [38].

TABLE I. EXPERIMENTAL INFRASTRUCTURE

Cluster	Nodes	CPU	Memory	Role
Orion	4	2x6 cores @2.30Ghz	32 GB	SeD
Sagittaire	38	2x1 core @2.40Ghz	2 GB	SeD
Taurus	10	2x6 cores @2.30Ghz	32 GB	SeD
Stremi	38	2x12 cores @1.70Ghz	48 GB	SeD
Graphite	4	2x6 cores @2.00Ghz	64 GB	SeD
Parasilo	17	2x6 cores @2.40Ghz	128 GB	SeD
Parasilo	1	2x6 cores @2.40Ghz	128 GB	MA
Parasilo	1	2x6 cores @2.40Ghz	128 GB	Client

B. Parallelization Investigations

We first investigate parallelization of the NSDE-2 algorithm on a handy 8-core Intel laptop with chipset i7-4710HQ@2.5Ghz. These are offline simulations using data for a set of 500 tasks to be placed on 85 servers, where task and server data have been extracted from the GRID'5000 testbed. A population size of 200 is considered for all simulations as well as online optimization executions.

The master-slave approach is followed in parallelization, using the *Open MP* Library. In the NSDE program the functions not amenable to parallelization include the selection operations using non-dominated sorting and crowding distance algorithms that take up about 3.3% of runtime, and some I/O operations taking approximately another 1%. It follows from Amdahl's law that the *Theoretical Maximum Speedup* factor is approximately 22.

Table 2 shows results obtained using the sequential NSDE-2 program, parallelized NSDE-2 program running on a single core, and on 4 cores. The data shows computation times and the average values over all candidates for the two objectives, energy and makespan, for a workflow of 500 tasks on 85 servers. The relevance of the average values in these multi-objective simulations is purely to check if the sequential and 1-core-parallel solutions match exactly, which they are observed to do. This, first and foremost, demonstrates the correctness of parallelization. Second, it shows that the speedup factor on 4 cores is 2.36.

An interesting observation from Table 2 is that the speed of evolution of candidates across generations varies between the parallel solutions and the sequential, reflected in different numerical values of the objectives at the same generation levels. It may be difficult to pinpoint the reasons for this; the evolutionary algorithm being a stochastic process is likely to behave differently when executed concurrently on different numbers of nodes, and these differences are likely to amplify over generations.

Figure 2 shows the parallelization speedup factor when running selected sets of 100 and 1000 tasks on 85 servers, real time on the GRID'5000 testbed on a Stremi node (see Table 1). It may be noted that in the NSDE solution framework, each

TABLE 2. SEQUENTIAL AND PARALLEL SIMULATIONS

Parallelization	Time for 3000 gens (mins)	Average value at 100 generations (Makespan in mins, Energy in kJ)		Average value at 3000 generations	
		Makespan	Energy	Makespan	Energy
Sequential	33:52	39.36	4491.4	37.05	3546.6
1-core	33:49	39.36	4491.4	37.05	3546.6
4-core	14:22	40.46	4874.8	35.85	4061.3

task is effectively a decision (design) variable, and obtaining optimized solution with 1000 variables is itself a challenging task. In fact, this

number has been extended to 5000 decision variables on 85 servers launched in parallel on 24 nodes, though comparative sequential runs could not be obtained due to runtime constraints. Fig. 2 shows that as the size of the workflow increases, the parallelization speedup factor gradually approaches its maximum limit.

B. Quality of NSDE-2 optimized workflow placements

Figure 3 plots the evolution of NSDE-2 solutions from an initial level of 100 generations up to 10000 generations, with minimization of energy consumption and makespan as the objectives on the two axes. Each dot represents a candidate solution. At any selected generation, at one end of the solution 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, and the "cloud" of solutions gradually transforms into a Pareto front. The computation time increases linearly with the number of generations. We choose to retrieve the solution at 3000 generations, after which the improvement in solution becomes less significant.

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

Fig. 2. Parallelization speedup factors for different job sizes and cores.

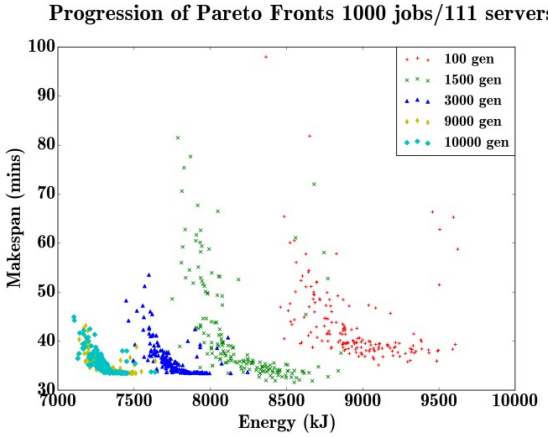


Fig. 3. Convergence towards a Pareto front across generations.

TABLE 3. COMPARATIVE IMPROVEMENTS IN MAKESPAN AND ENERGY USING NSDE-2 (figs. show % gains against FIRST FIT)

Cases	No. of jobs	NSDE-2 Computation time (mins)	Make span for NSDE-2 Best Make span (%)	Make span for NSDE-2 Best Energy	Energy for NSDE-2 Best Make span	Energy for NSDE-2 Best 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

Next we compare the distribution of tasks among nodes on GRID'5000 attained under 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 candidate solutions on the same Pareto front with the smallest energy consumption and the smallest makespan. These solutions establish the bounds of the Pareto Front. The FIRST FIT policy selects the first available server in an ordered list according to the *GreenPerf* metric as a non-weighted average ratio between makespan and energy consumption for the said type of task.

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 by computing a sample of each type of task on the servers prior to initiation of the evolutionary optimization process.

Figures 4-7 show the results of these studies on 100, 200, 500 and 1000 jobs launched on 111 servers. The x-axis presents the different policies 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 that as the complexity and size of the optimization space increases (cf. Sec IVa where the size varies as m^n), the multi-objective evolutionary algorithm provides better solutions compared to the single-metric based ranking approach of FIRST FIT. This is also reflected in Table 3. It may be seen that energy consumption improves in all NSDE scenarios (best-energy and best-makespan) up to 25%, while makespan improves for the best-makespan solution for the larger cases. Further, NSDE provides a spectrum of intermediate solutions to the user to select based on his preferences between energy and makespan.

When the NSDE-2 solution was run up to 10000 generations, it provided a 30% saving in energy with a 50% reduction in makespan without considering the algorithm convergence time. This can be of value in cases of jobs submitted by prior reservation, as typical of jobs involving large computation times.

CONCLUSIONS AND PERSPECTIVES

This work describes the design, implementation and evaluation of an energy-efficient resource management system that builds upon DIET, an open source middleware and NSDE-

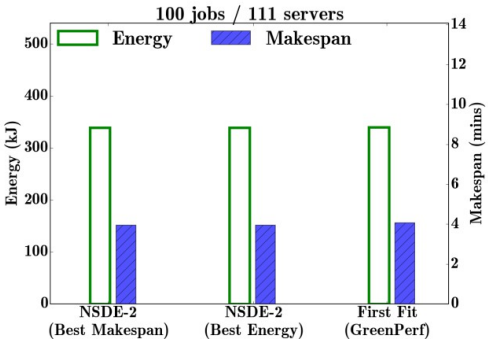


Fig. 4. Energy and Makespan comparison for 100 jobs and 111 servers.

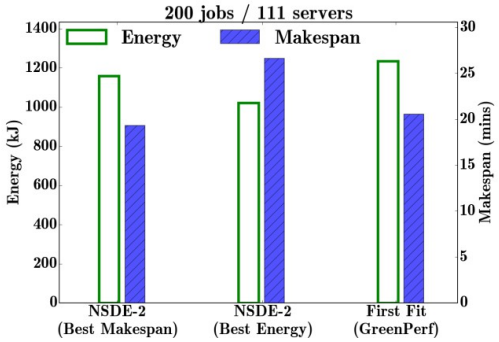


Fig. 5. Energy and Makespan comparison for 200 jobs and 111 servers.

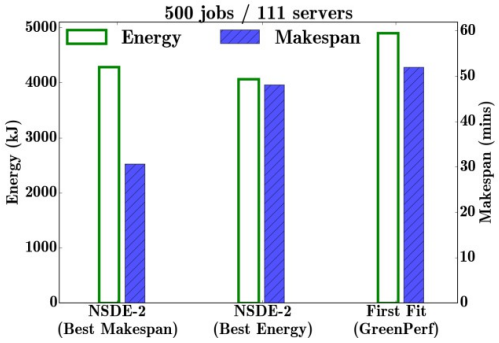
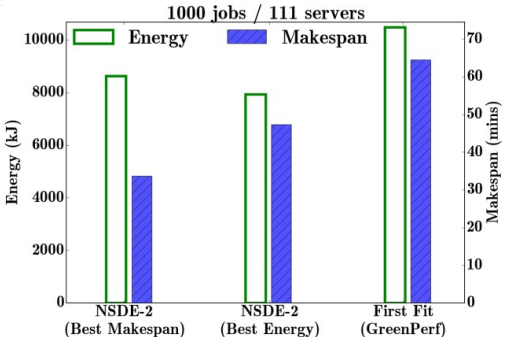


Fig. 6. Ener



111 servers.

Fig. 7. Energy and Makespan comparison for 1000 jobs and 111 servers.

II, an Evolutionary Multi-Objective Optimization engine. It investigates the nuances of parallel launching of the computation-intensive evolutionary algorithm on the baseline cloud infrastructure where the workflow is targeted for optimal placement. The implementation supports an IaaS Cloud and currently provides placement of workflows, considering non-divisible tasks with precedence constraints. Real-life experiment of this approach on the GRID'5000 testbed demonstrates its effectiveness in a dynamic environment. Results shows that our method can offer providers and decision makers an aid to make decisions when conflicting objectives are present, or when in search for realistic trade-offs for a given problem.

Investigations on parallelization of NSDE-2 show that speedup values approaching the

theoretical maximum limit have been obtained on a Cloud cluster. It is observed that speed of evolution of solutions across generations fluctuates with the number of active cores. At the next stage, we will investigate multi-core integration of servers in a cluster, which is expected to yield significant energy savings.

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.C. Orgerie, M. D. d. Assuncao, and L. Lef`evre, "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.

[4] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic onsolidation 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.

[5] A.-C. Orgerie and L. Lef`evre, "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

[6] 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; IPDPS 2015, May 2015.

[7] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms." in *Proc. First International Conference on Genetic Algorithms*, Pittsburgh, PA, USA, July 1985, pp. 93–100.

[8] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA-2: Improved performance of the Strength Pareto Evolutionary Algorithm," *Technical Report 103*, Computer Engineering and Communication Networks Lac (TIK), Swiss Federal institute of Technology (ETH) Zurich, 2001.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist Multiobjective Genetic Algorithm: NSGA-II," IEEE Trans. on Evolutionary Computation, vol. 6, no. 2, 2002, pp. 182–197.

[10] J. Lampinen, "Differential evolution: New naturally parallel approach for engineering design optimization," in *Developments in Computational Mechanics with High Performance Computing*, B. H. V. Topping, Ed. Edinburgh, U.K.: Civil-Comp Press, 1999, pp. 217–228.

[11] D.K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel differential evolution," in *Proc. Congr. Evol. Comput.*, 2004, pp. 2023–2029.

[12] E.G. Talbi et al, "Parallel Approaches for Multi-Objective Optimi-zation", LNCS 5252, Eds. J. Branke et al, Springer 2008, pp. 349-372.

[13] J. H. Holland, "*Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*", Univ. of Michigan Press, 1975.

[14] R. Storn and K. Price, "Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces," Journal of global optimization, vol. 11, no. 4, 1997, pp. 341–359.

[15] A.K. Bhattacharya, S. Debjani, A. RoyChoudhury and J. Das, "Optimization of Continuous Casting Mould Oscillation Parameters in Steel Manufacturing Process using Genetic Algorithms", Proc. IEEE Congress on Evol. Com., CEC2007, pp. 3998-4004, Sep 25-28, 2007.

[16] 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, 2013, pp. 2767–2789.

[17] 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, pp 77-102.

[18] W. Lee, Y. Hsiao and W. Hwang, "Designing a parallel evolutionary algorithm for inferring gene networks on the Cloud computing environ-ment", BMC Systems Biology, DOI: 10.1186/1752-0509-8-5 2014.

[19] C. Deng, X. Tan, X. Dong and Y. Tan, "A parallel version of Differential Evolution based on Resilient Distributed Datasets model", Comms. in Computer and Info. Science, 562, Springer 2015, pp. 84-93.

[20] 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, 2006, pp. 335–352.

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

[22] 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, 2013, pp. 3045–3055.

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

[24] F. Legillon, N. Melab, D. Renard, and E.-G. Talbi, "A multi-objective evolutionary algorithm for cloud platform reconfiguration," Parll. and Distrib. Processing Symp. WS, IEEE, 2015, pp. 286–291.

[25] A. Benyi, J. D. Dombi, and A. Kertesz, "Energy-aware VM scheduling in IaaS clouds using pliant logic," in Proc. 4th Int. Conf. on Cloud Computing and Services Science (CLOSER14), Barcelona, Spain, 2014.

[26] M. Mezmaz et al, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems", Parallel and Distrib. Comp., Vol. 71, 2011, pp. 1497-1508.

[27] C. Tsai and J.J. Rodrigues, "Metaheuristic Scheduling for Cloud: A Survey", IEEE Systems Journal, Vol. 8, No. 1, 2014, pp. 279-291.

[28] K. H. Kim, R. Buyya, and J. Kim, "Power aware scheduling of bag-oftasks applications with deadline

constraints on DVS-enabled clusters," in CCGRID, IEEE Computer Society, 2007, pp. 541–548.

[29] 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.

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

[31] D. Wentzlaff et al., "An operating system for multicore and clouds: Mechanisms and implementation," in Proc. 1st ACM Symposium on Cloud Computing, SoCC '10. New York, ACM, 2010, pp. 3–14.

[32] 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. 2nd Int. Workshop on Green Computing, IEEE, Sep. 2010, pp. 497–503.

[33] E. Caron, B. Depardon, and F. Desprez, "Deployment of a hierarchical middleware," in Euro-Par 2010, LNCS. 6271 Part I. Ischia - Naples, Aug-Sep 2010, pp. 343–354.

[34] S. Das and P. N. Suganthan, "Differential Evolution: a survey of the state-of-the-art," IEEE Trans. on Evol. Comp., vol. 15, 2011, pp. 4–31.

[35] F. Cappello et al, "Grid'5000: a large scale, reconfigurable, controllable and monitorable Grid platform," in Proc. 6th IEEE/ACM Int. Workshop on Grid Computing, Grid'2005, Seattle, Nov. 2005, pp. 99–106.

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

[37] M. D. Assuncao, J.P. Gelas, L. Lef`evre, and A.C. Orgerie, "The green grid'5000: Instrumenting and using a grid with energy sensors," Remote Instr. for eScience and Related Aspects, Springer, 2012, pp. 25-42.

[38] D. Balouek, A.K. Bhattacharya, E. Caron, K. Gadireddy and L. Lefevre, "Minimizing Energy and Makespan concurrently in Cloud Computing workloads using Multi-Objective Differential Evolution", unpublished, submitted ICDCS 2016.