A Power Efficient Genetic Algorithm for Resource Allocation in Cloud Computing Data Centers

Giuseppe Portaluri
Universitá di Pisa
giuseppe-portaluri@libero.it

Stefano Giordano
Universitá di Pisa
s.giordano@iet.unipi.it

Dzmitry Kliazovich
University of Luxembourg
dzmitry.kliazovich@uni.lu

Bernabé Dorronsoro
University of Luxembourg
bernabe.dorronsoro@uni.lu

***Abstract*—One of the main challenges in cloud computing is to increase the availability of computational resources, while minimizing system power consumption and operational expenses. This article introduces a power efficient resource allocation algorithm for tasks in cloud computing data centers. The developed approach is based on genetic algorithms which ensure performance and scalability to millions of tasks. Resource allocation is performed taking into account computational and networking requirements of tasks and optimizes task completion time and data center power consumption. The evaluation results, obtained using a dedicated open source genetic multi-objective framework called jMetal show that the developed approach is able to perform the static allocation of a large number of independent tasks on homogeneous single-core servers within the same data center with a quadratic time complexity.**

***Index Terms*—Data center, Cloud Computing, Genetic Algorithm, Resource Allocation, Power Efficiency.**

I. Introduction

Cloud computing data centers provide high performance, scalability and access to virtually unlimited computational power to application providers. For optimal operation, data centers need to perform hardware maintenance, provide redundancy, optimize power consumption as well as manage task execution and network traffic. Energy and power consumption in data centers affect operational costs. In recent years, different solutions where developed to improve energy efficiency of computing hardware and network equipment.

VMPlanner [1] is a VM and traffic flows allocator able to reduce data center power cost by putting in sleep mode network elements. VMPlanner solves the Virtual Machine (VM) allocation problem with three different algorithms which use approximation and are not scalable due to only a limited number of switches and racks that could be considered.

In [14] authors developed a particular kind of Service Level Agreement (SLA) called Green SLA. Green SLA uses best effort scheduling, which minimizes task execution time and energy-performance trade off. This approach implements a number of advanced power management strategies such as Dynamic Voltage and Frequency Scaling (DVFS) and supports parallel execution.

In [2] the authors propose another energy efficient dynamic allocator for VMs. It is implemented using a modified version of the Best Fit Decreasing algorithm to allocate VMs using their utilization factors. This approach allows to dynamically reallocate VMs and supports also heterogeneous hardware.

Another approach, presented in [4], is able to allocate tasks with the objectives of minimizing the longest task completion time while optimizing the energy efficiency at the same time in heterogeneous Grid systems composed of multi-core processors. However, that work does not take into account the network requirements.

In [16], the authors propose a system which maps VMs to physical resources using genetic algorithm improved with fuzzy multi-objective optimization. This approach tries to reduce the amount of power consumed by the servers, optimize CPU and memory utilization, and minimize peak temperatures inside the facility. In summary, most of the reviewed genetic algorithm based approaches optimize only one or two parameters at a time and primarily focus on computing hardware, while all parameters including system performance, networking requirements, completion time of the tasks and energy consumption must be taken into account.

In this paper, we propose a new resource allocation approach for cloud computing data centers that performs joint allocation of computational and network resources. Our objective is finding trade-off solutions between tasks completion time and system power consumption. The system considered includes the static scheduling of independent tasks on homogeneous single-core resources. This algorithm is designed using genetic algorithms that allow both to explore solutions space and to search for the optimal solution in an efficient manner. It is both scalable and power efficient, and is based on a model developed to capture specifics of the data center network topology and device power consumption. During the implementation phase, a specific java-based open source framework for multi-objective genetic algorithms, called jMetal [6], was used.

The rest of the paper is organized as follows: Section II defines the problem and the proposed system model, Section III details the developed algorithm, Section IV includes a description of the performed experiments and analyses the obtained results, Section V concludes the paper outlining directions for future research on the topic.

II. System Model

One of the main challenges in cloud computer industry is data center energy efficiency and scalability. Total power consumption due to data center, in a country such as US, doubled in 6 years (from 2000 to 2006) reaching nearly 61 billion kW h or 1.5% of total US electricity consumption[1].

978-1-4799-2730-2/14/$31.00 ©2014 IEEE

For every watt delivered [13] only 30% is consumed by the IT equipment, 33% are spent in chillers, 18% in uninterruptible power suppliers, 9% for computer room air conditioning, 5% in power distribution units, 3% in humidifiers. 1% for lightning and 1% in transformers.

There are two main approaches for power management in server hardware: DVFS and dynamic power shutdown. DVFS reduces power consumption by lowering down operating frequency and/or voltage, while dynamic power shutdown saves power during idle times by powering down as much as possible all the sub-components. The aforementioned approaches can be applied in both computing hardware and network switches. Network operators can free some devices by reducing the number of used paths and aggregating the traffic. This approach can be especially beneficial as average utilization of network links is often below 40%.

We model data center with currently the most widely used three-tier fat-tree architecture (see Fig. 1). IT is composed of the access, aggregation, and core layers. The access layer provides connection to servers which are arranged into racks with each rack being served by a single *Top of the rack* (ToR) switch. In our model, one rack contains of up to 24 servers and a single pod includes up to 8 racks.

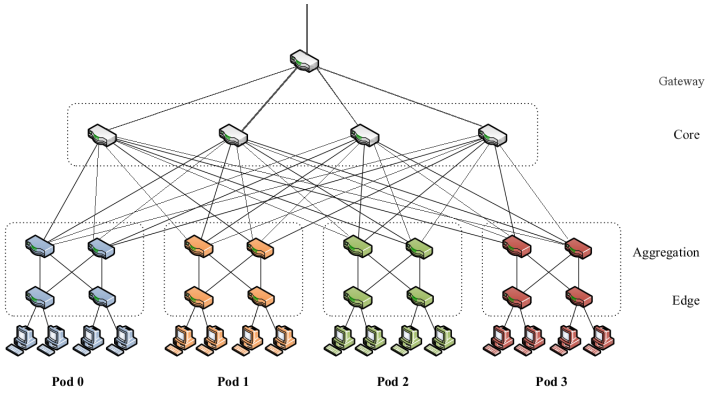

Fig. 1. Fat-tree topology in data centers.

The links interconnecting server and ToR switches have capacity of 1 Gb/s while the links between racks and aggregation switches are 2 x 10 Gb/s. The proposed algorithm allocates in this data center, which is empty at the beginning, a set of independent *tasks* which are characterized by:

- a number of instructions to be executed;
- a constant amount of bandwidth required to perform the execution.

As all tasks are independent and do not require to communicate during their execution. For this, the bandwidth requirement is only related to communication between computing servers and data center gateway.

Computing servers are modeled with single-core processors offering a fixed computational power expressed in instructions per second. The tasks allocated for execution on the same server will share server's processing power equally. The servers left idle can be put into a sleep mode minimizing their power consumption. As a result, the power model of data center network is linearly proportional with the respect to the load. Power consumption ranges between two power values: idle and peak power respectively when traffic load is 0 or 1. If dynamic power shutdown is enabled, the servers and network switches that left idle are turned off.

Considering:

N tasks to be allocated
M servers
S switches
R racks
IPS instruction per second executed by CPUs
n_j tasks allocated on j-th server
P_{SRP} peak power consumption for servers
P_{SWP} peak power consumption for switches
P_{SWI} idle power consumption for switches

we define

T_c the maximum completion time between all the tasks
B_i the i-th task bandwidth request
P_{SRj} the power consumption of server j-th
$P_{SR} = \sum_j P_{SRj}$ the total server power consumption
P_{SWk} the power consumption of k-th switch
$P_{SW} = \sum_k P_{SWk}$ the total switch power consumption
$P = P_{SR} + P_{SW}$ the global power consumption
V_{Lj} the bandwidth utilization on j-th server link
V_{Tk} the bandwidth utilization on k-th ToR switch

The following two objectives must be minimized:

1) Maximum total completion time of all tasks (*makespan*);
2) Power consumption of servers and network switches.

The first objective represents performance of the whole data center, while the second objective is related to its power consumption. Our goal is to find a reasonable trade off. These two objectives can be represented by the following two fitness functions:

$$\begin{cases} f_1 = T_c; \\ f_2 = P. \end{cases} \tag{1}$$

The total completion time corresponds to the completion time of the last task.
Considering

$x_{ij} = \begin{cases} 1, \text{ if the i-th task is allocated on the j-th server;} \\ 0, \text{ otherwise;} \end{cases}$
$\#ins_i$ the total number of instruction of task i-th;
$I_j = \sum_{i=1}^{N} x_{ij} \ \#ins_i$ instructions to be executed on j-th server
$T_j = \frac{I_j}{IPS}$ the j-th server completion time

we can define the total completion time as

$$T_c = max_{\ 1 \le j \le M} \ T_j. \tag{2}$$

The power consumption of computing servers is modeled using binary law, assuming that each processor either execut-

ing tasks at the full speed or stays idle:

$$P_{SRj} = \begin{cases} P_{SRP}, & \text{if } n_j \geq 1; \\ 0 & \text{if } n_j = 0. \end{cases} \quad (3)$$

The power consumption of a network switch is linearly proportional to the traffic load and stays between P_{SWP} (when all the link are full utilized) and P_{SRI} (when the switch is idle). Network switches left idle can be turned off to optimize their energy consumption. Defining

C_i the throughput of the i-th link
C_{MAX} the sum of maximum capacity of all uplinks

$\#link_k$ the number of uplinks in switch k-th
$l_k = \sum_{i=1}^{\#link_k} \frac{C_i}{C_{MAX}}$ the load factor of k-th switch

the network switch power consumption can be computed as follows:

$$P_{SWk}(l_k) = \begin{cases} P_{SWI} + (P_{SWP} - P_{SWI}) l_k & \text{if } 0 < l_k \leq 1; \\ 0 & \text{if } l_k = 0. \end{cases} \quad (4)$$

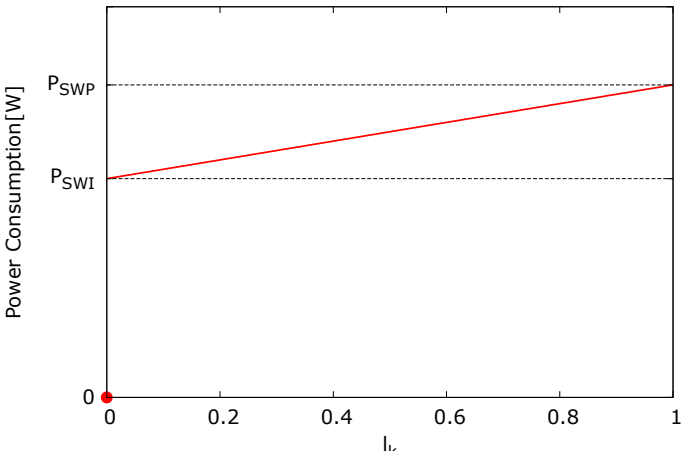

Fig. 2. Network switch power consumption.

The optimization problem is the subject to two different constraints: it is not possible to allocate on a single server tasks demanding higher bandwidth than the available link capacity (1Gb/s), and the 20 Gb/s uplink capacity of ToR switches cannot be exceeded by the sum of the traffic. These two constraints can be formalized as follows:

$$\begin{cases} V_{Lj} \leq 1Gb/s & 1 \leq j \leq M; \\ V_{Tk} \leq 20Gb/s & 1 \leq k \leq R. \end{cases} \quad (5)$$

Denoting

$x_{ijk} = \begin{cases} 1, \text{ if the i-th task is allocated on the j-th server of the k-th rack;} \\ 0, \text{ otherwise;} \end{cases}$

$$c_{j1} = \left(\sum_{i=1}^{N} x_{ij} \, B_i - 1Gbps \right) . sgn \left(\sum_{i=1}^{N} x_{ij} \, B_i - 1Gbps \right)$$

$$c_{k2} = \left(\sum_{j=1}^{M} \sum_{i=1}^{N} x_{ijk} \, B_i - 20 \, Gbps \right) sgn \left(\sum_{j=1}^{M} \sum_{i=1}^{N} x_{ijk} \, B_i - 20 \, Gbps \right)$$

the two constraints are:

$$\begin{cases} c_1 = \sum_{j=1}^{M} c_{j1}; \\ c_2 = \sum_{k=1}^{R} c_{k2}. \end{cases} \quad (6)$$

where

$$sgn(x) = \begin{cases} 0 \; \forall x \leq 0; \\ 1 \; \forall x > 0. \end{cases} \quad (7)$$

The aforementioned definition of two constraints does not assign any penalty if network links are not congested. Whenever tasks allocated to the same server (or to the same rack) require more than available bandwidth, the exceeding amount of bandwidth is added to the constraint violations. It is important to recall that with this heuristic, solutions that are violating constraints are still considered during the search phase, but using the proper search heuristic they tend to disappear in the last generations.

III. Optimal Allocation of Resources using NSGA-II

Genetic algorithms (GAs) [15] are iterative stochastic optimization methods based on the principles of natural selection and evolution. In GAs, a population of candidate solutions (called individuals or phenotypes) is evolved toward better solutions with the application of genetic operators. Each solution has a set of properties (called chromosomes or genotype) which can be recombined, randomly mutated and altered to form a new set of solutions named generation. In each iteration the population of solutions is evaluated considering the fitness of every individuals (usually a value related to the objective function). The more fit individuals are selected and survive for consequent iterations. In multi-objective algorithms, solutions are not comparable because of the problem of incommensurability. For this reason it is introduced the concept of *Pareto-optimal solutions*. Consider two solutions $\mathbf{x}$ and $\mathbf{y}$ and a set of objectives that should be maximized $f_1, f_2, ..., f_m$, $\mathbf{x}$ *dominates* $\mathbf{y}$ if exists a value k such that:

$$\begin{cases} f_k(\mathbf{x}) > f_k(\mathbf{y}); \\ f_l(\mathbf{x}) \geq f_l(\mathbf{y}) \quad 1 \leq l \leq m \wedge l \neq k. \end{cases} \quad (8)$$

If that value of k does not exist, $\mathbf{y}$ is *non-dominated* by $\mathbf{x}$.

The main task of a multi-objective GA is to find a set of *non-dominated* solutions (or Pareto-optimal solutions) optimizing the current problem.

Non-dominated Sorting Genetic Algorithm II (NSGA-II) [8] is a widely adopted algorithm for solving multi-objective optimization problems. NSGA-II has a complexity of $O(MN^2)$ with M the number of objectives and N the population size. This heuristic sorts the population of solutions into different non-domination levels with a procedure called *ranking*: if a solution p dominates a solution q, then p belongs to a higher

level than q. This procedure is repeated for every solution creating different groups or *non-domination levels* (solutions of the same group are non-dominating themselves); an integer value called *rank* is assigned to each non-domination level (1 is the best level, 2 is the second best level, and so on). When applying selection and sorting, NSGA-II is able to deal with constraints and unfeasible solutions. The alternatives in comparing two solutions are:

- both solutions are feasible and the one with the best fitness is chosen;
- only one solution is feasible and that one is chosen;
- both solutions are unfeasible and the one with the smallest overall constraint violation is chosen.

Using multi-objective optimization, it is necessary to introduce the definition of *constraint domination*: a solution i is said to constrained-dominate a solution j, if any of the following conditions is true:

1) solution i is feasible and solution j is not;
2) solutions i and j are both unfeasible, but solution i has a smaller overall constraint violation;
3) solutions i and j are feasible and solution i dominates solution j.

All feasible solutions are ranked according to their non-domination level which is based on the objective function values. Among two unfeasible solutions, the solution with a smaller constraint violation has a better rank. When the ranking procedure is carried out, a number of chromosomes equal to the population size is taken from the best ranked solutions. If adding a rank of non-dominated solutions implies exceeding the population size, only a subset of solutions from this rank are chosen. These solutions are chosen according to the crowding distance method. The crowding distance value of a solution provides an estimate of the density of solutions surrounding it. Crowding distance of point i is an estimate of the size of the largest cuboid enclosing i without including any other point (Fig. 3). Boundary solutions which have

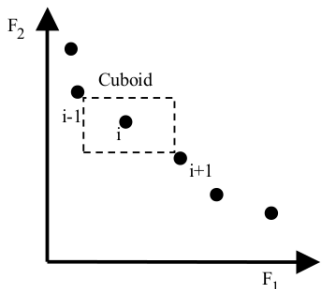

Fig. 3. Crowding distance for i-th solution in a two objectives algorithm.

the lowest and highest objective function values are given an infinite crowding distance. Solution A is better ranked than solution B if and only if:

$$non-dominated\ level(A) < non-dominated\ level(B). \quad (9)$$

or

$$\begin{cases} non-dominated\ level(A) = non-dominated\ level(B); \\ crowding\ distance(A) > crowding\ distance(B). \end{cases} \quad (10)$$

Once the new population is built, it undergoes the application of the selection and genetic operators to generate a number of new solutions, from which the new population will be built for the next iteration.

IV. Experimental setup

In this section we describe the executed tests with the developed algorithm and the obtained results.

A. Scenario

Data center topology, tasks, servers and values for hardware power consumption are described in Tables I and II, while the algorithm configuration setup is presented in Table III. The number of servers per rack and rack per pod are typical values for a data center [7], the number of instructions per second (IPS) is frequent in single core processors such as Intel Pentium at 1 GHz, the task instruction number and bandwidth requirement for each task are generated randomly using two different uniform distributions. In Table II, power consumption values are reported; for switches idle power consumption is considered to be the 80% of the peak power consumption [7]. In Table III, we used the original parameters proposed for NSGA-II, and we implemented two classical Crossover and Mutation operators for combinatorial problems like ours (NSGAII was designed for continuous problems). We encode solutions as an integer vector of size equal to the number of tasks. The value assigned to every position of the chromosome represents a server where the corresponding task is allocated to (this value ranges from 1 to the number of servers).

TABLE I
Topology, server and task related parameters

Parameter	Value
Server per rack	24
Rack per pod	8
IPS	$1,2 * 10^9 [instr/sec]$
Task instruction distribution	Uniform in the range $[5; 10] * 10^9$ $[instr]$
Task bandwidth distribution	Uniform in the range $[1; 1000] * 10^8$ $[bps]$

TABLE II
Power consumptions

Parameter	Value [W]
Server P_{SRP}	300
ToR P_{SWP}	200
ToR P_{SWI}	160
Aggregation P_{SWP}	2500
Aggregation P_{SWI}	2000

B. Results

Three different experiments were performed. In the first experiment, we analyse the best solutions taking into account the two objectives separately in order to explore the two edges

TABLE III
GENETIC ALGORITHM PARAMETERS

Parameter	Value
Population size	100
Evaluations number	25000
Crossover operator	One point
Crossover probability	0.9
Mutation operator	Random mutation
Mutation probability	$\frac{1}{\text{task number}}$
Selection operator	Binary tournament
Search heuristic	NSGA-II

of the obtained Pareto front. The second experiment shows an example of Pareto front obtained and a superimposition of different Pareto fronts obtained through different experiments using the number of servers as a parameter of interest. Finally, the third experiment shows the experimental time complexity as a function of the number of tasks to allocate. The platform used to execute these experiments is an Intel I7 3630QM 2.4 GHz equipped with 8 GB RAM with OS Ubuntu 11.04 and the jMetal framework.

1) Time and Power consumption: The first experiment was executed with input values presented in Table IV.

TABLE IV
EXPERIMENT 1: INPUT VALUES

Parameter	Value [W]
Server number	1536
Task number	[500:15000] with steps of 500
Same experiment repetition	40

Figure 4 shows solutions with optimal completion time. As expected, the observed trend is linear with respect to the

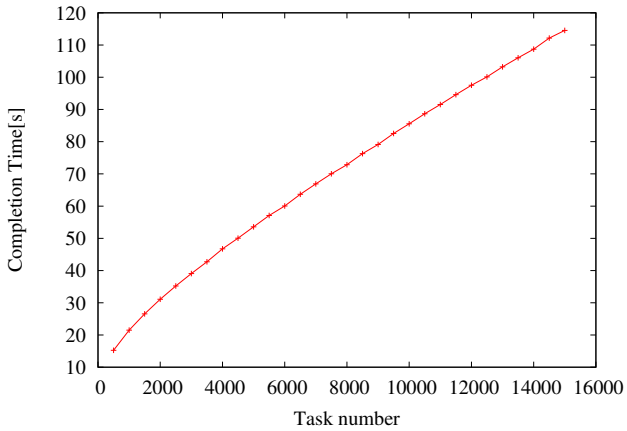

Fig. 4. Best task completion times.

number of tasks. This set of solutions represent the best time-performant solutions found by the algorithm. We are looking now also for the most power performant solutions within the same experiment. Figure 5 shows the best values of power consumption retrieved using the same technique as before to compute best completion times. Here the curve follows a

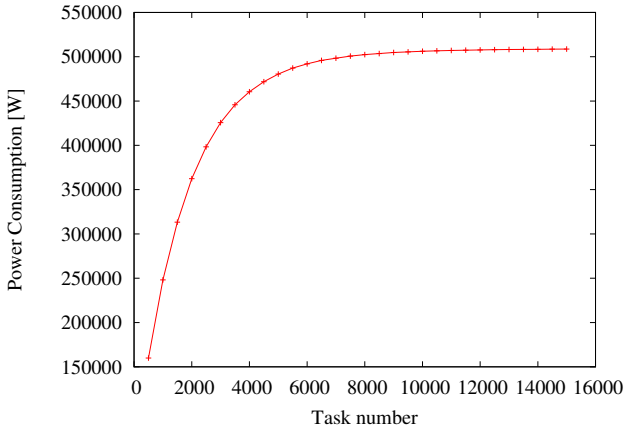

Fig. 5. Best power consumption values.

different trend because if the number of server and switches is constant, after increasing the number of tasks the final effect is to reach the limit of the sum of the peak power consumption for every devices into the data center.

2) Pareto-optimal solutions: this experiment shows the relation between solutions of different executions of the problem for varying server numbers.

Figure 6 shows the Pareto front obtained by the algorithm for a single instance of 1536 servers and 3000 tasks. In it the completion time ranges from around 39 seconds to 51 seconds (best found solutions) while the power consumption is between 425 KW to 443 KW. If power consumption and completion

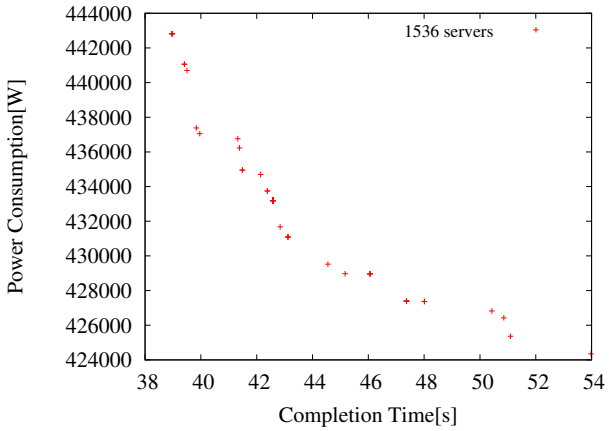

Fig. 6. Pareto front obtained by a single experiment.

time could be considered equally relevant the vector with minimal module can be considered as the best solution. On the contrary, if power consumption (or completion time) would be more relevant the two metrics could be weighted to find the preferred solution.

Figure 7 shows the superimposition of solutions varying the number of servers; each execution is repeated forty times and we plot the forty obtained Pareto fronts.

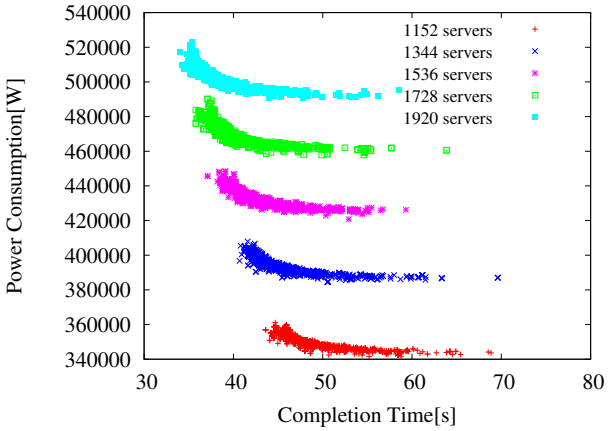

Fig. 7. Solution with different number of servers.

We can observe how increasing the number of servers, leads to a deprecation of the results obtained in terms of power consumption and completion time, as it was expected.

3) Algorithm execution time: last experiment measures the performance of the framework to compute the solution with a fixed number of servers (i.e.1536) usually constant for a specific data center, and varying the task number which is the only variable input parameter. Figure 8 shows the results with a trend line. Points are close to the quadratic trend function

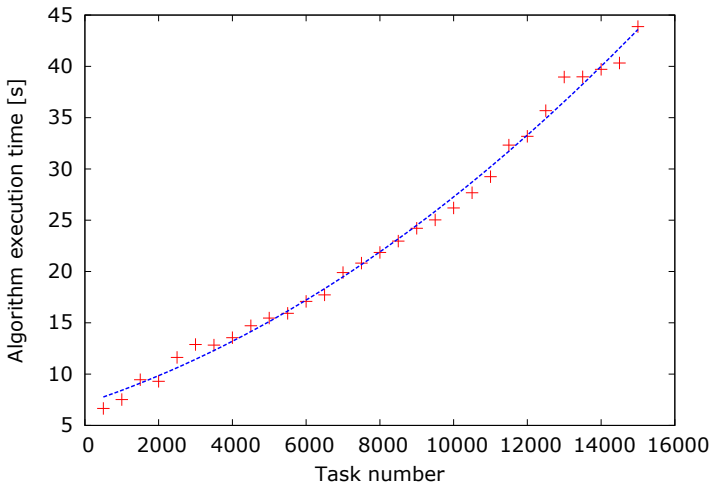

Fig. 8. Execution time of the algorithm.

that fits well the measured values then the algorithm in the considered range of values has a quadratic dependence from the number of input tasks.

V. Conclusions and Future Work

In this paper we introduce a power efficient resource allocation algorithm for cloud computing data centers which is based on genetic heuristics. The proposed approach finds a set of non-dominated solutions in this multi-objective computation minimizing makespan and power consumption of the system. When the execution of the algorithm is completed and optimal Pareto solutions are obtained, it becomes is possible to fine tune the trade-off between power consumption and execution time. The proposed algorithm shows quadratic complexity dependency on with the respect to the number of tasks to be allocated.

Future work will focus on model adaptation to dynamic task allocation, account for internal communications, electricity cost and data center load factor and provide comparisons with other state-of-the-art algorithms.

Acknowledgment

The authors would like to acknowledge the funding from National Research Fund, Luxembourg in the framework of ECO-CLOUD project (C12/IS/3977641) and AFR contract no. 4017742

References

[1] Weiwei Fang, Xiangmin Liang, Shengxin Li, Luca Chiaraviglio, Naixue Xiong, "VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers", Computer Networks, Volume 57, Issue 1, 16 January 2013, Pages 179-196

[2] Anton Beloglazov, Rajkumar Buyya, "Energy Efficient Allocation of Virtual Machines in Cloud Data Centers", Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010

[3] Fatos Xhafa, Enrique Alba, Bernabé Dorronsoro, "Efficient Batch Job Scheduling in Grids using Cellular Memetic Algorithms", Parallel and Distributed Processing Symposium, Pages 1-8, 26-30 March 2007.

[4] Sergio Nesmachnow, Bernabé Dorronsoro, Johnatan E. Pecero, Pascal Bouvry, "Energy-aware scheduling on multicore heterogeneous grid computing systems", Journal of Grid Computing, Vol. 11, Issue 4, 2013, pages 653-680.

[5] Mitsuo Gen, Runwei Cheng, Lin Lin, "Network Models and Optimization, Multiobjective Genetic Algorithm Approach", Springer, 2008.

[6] Juan J. Durillo, Antonio J. Nebro, "JMetal: A Java framework for multi-objective optimization", Advances in Engineering Software, Elsevier, Volume 42, Issue 10, October 2011.

[7] Dejene Boru, Dzmitry Kliazovich, Fabrizio Granelli, Pascal Bouvry, Albert Zomaya, "Energy-Efficient Data Replication in Cloud Computing Datacenters", CCSNA workshop at Globecom, December 2013.

[8] Kalyanmoy Deb, Amrit Pratap, Samee Agarwal, T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", IEEE Transactions on Evolutionary Computation, April 2012

[9] Carlo R. Raquel, Prospero C. Naval jr, "An Effective Use of Crowding Distance in Multiobjective Particle Swarm Optimization", Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, 2005, Pages 257-264

[10] Albert Greenberg, James Hamilton, David A. Maltz, Perveen Patel, "The Cost of a Cloud: Research Problems in Data Center Networks", SIGCOMM Comput. Commun. Rev., January 2009

[11] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, Songwu Lu, "DCell: "A Scalable and Fault-Tolerant Network Structure for Data Centers", Proceedings of Sigcomm 2008

[12] Mysore, Pamboris, Farrington, Huang, Miri, Radhakrishnan, Subramanya, Vahdat, "Portland: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric", Sigcomm 2009

[13] Luiz A. Barroso, Jimmy Clidaras, Urs Hlzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines", Synthesis Lectures on Computer Architecture 8(3), Pages 1154, 2013

[14] Lizhe Wang, Gregor von Laszewski, Jai Dayal, Fugang Wang, "Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS", 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010

[15] Goldberg, David E., "Genetic Algorithms in Search, Optimization and Machine Learning",Addison-Wesley Longman Publishing, 1989.

[16] Jing Xu, José A B Fortes, "Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments", Green Computing and Communications (GreenCom), Pages 179-188, December 2010.