# Energy-aware Grid Scheduling of Independent Tasks and Highly Distributed Data

Joanna Kołodziej
Cracow University of Technology
Cracow, Poland
jokolodziej@pk.edu.pl

Magdalena Szmajduch
CDN Partner Cracow
Poland
magdalena.szmajduch@cdnpartner.pl

Tahir Maqsood
COMSATS Institute of Information Technology (IT)
Pakistan
tmaqsood@ciit.net.pk

Sajjad A. Madani
COMSATS Institute of IT
Pakistan
madani@ciit.net.pk

Nasro Min-Allah
COMSATS Institute of IT
Pakistan
nasar@comsats.edu.pk

Samee U. Khan
North Dakota State University
Fargo, ND, 58108–5060, USA
samee.khan@ndsu.edu

***Abstract*—Data-aware scheduling in today's large-scale computing systems has become a major complex research issue. This problem becomes even more challenging when data is stored and accessed from many highly distributed servers and energy-efficiency is treated as a main scheduling objective. In this paper we approach the independent batch scheduling in grid environment as a bi-objective minimization problem with makespan and energy consumption as the scheduling criteria. We used the Dynamic Voltage and Frequency Scaling (DVFS) model for reducing the cumulative power energy utilized by the system resources for tasks executions. We developed for data transmission a general logical network topology and policy based on the sleep link-based Adaptive Link Rate (ALR) on/off technique. Two developed energy-aware grid schedulers are based on genetic algorithms (GAs) frameworks with elitist and struggle replacement mechanisms and were empirically evaluated for four grid size scenarios in static and dynamic modes. The simulation results show that the proposed schedulers perform to a level that is sufficient to maintain the desired quality levels.**

***Index Terms*—Energy Utilization; Green Computing; Data Grid; Scheduling; Data Center; Genetic Algorithm**

## I. Introduction

Today's grid and cloud systems are growing global-scale infrastructures enabling the remote access to the data and applications for High Performance Computing (HPC) users on demand from various locations. This growing demand the energy consumed by the clouds' and grids' datacenters that require high energy utilization to maintain data storage and processing, various types of services and users' computing applications. Data- and computation-intensive scheduling problems have been considered in a significant volume of the research work [6], [11]. However, they are usually addressed separately as different instances of a general grid scheduling problem. Most of the existing grid and cloud energy-aware scheduling methods target to save the energy consumed by single server, small computing cluster or one central datacenter. Dynamic Voltage and Frequency Scaling (DVFS) [2] method is usually applied in such approaches for the modulation of the power supply of the computing and data servers [5]. The idea of distributed data servers, or many distributed datacenters, which can be considered as components of large-scale distributed systems, is quite new but promising research direction is today's cluster and cloud computing [3], [15].

The main aim of this paper is to develop a scalable unified scheduling model, which can combine in the intelligent way two major system management and resource allocation mechanisms, namely computational-intensive and data-intensive models. Fig. 1 illustrates the correlation between those two models.

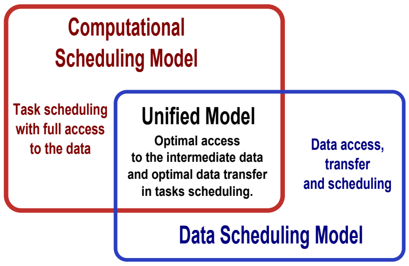
Computational Scheduling Model
Task scheduling with full access to the data
Unified Model
Optimal access to the intermediate data and optimal data transfer in tasks scheduling.
Data access, transfer and scheduling
Data Scheduling Model

Fig. 1. Commonality between data-aware and computational scheduling models

We address a general problem of batch scheduling in computational grids, where tasks are submitted to the system independently by the grid end-users. We focus on the global level of scheduling and propose a generic model of the management of highly distributed grid applications, data and resources. In this model, the grid computing nodes can be defined as single CPU machines, parallel multiprocessors and multicore machines or even small computing clusters. Data nodes can be considered as single data servers or distributed datacenters. The term 'task' may refer to a simple monolithic application or meta-task with several subtasks with some internal dependencies and relations, which require multiple data sets from different heterogeneous data hosts. These data sets may be replicated at various locations and can be transferred to the computational grid through the networks of various capabilities. A possible variant of this scenario is presented in Fig. 2.

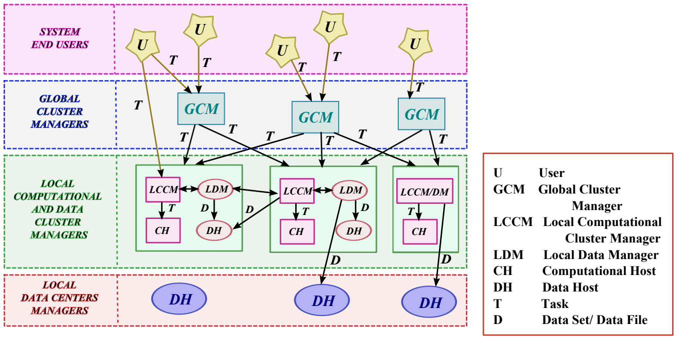

Fig. 2. Data-aware meta-task grid scheduling problem

It is assumed that each computational node can work at different ‘power supply levels’, which are defined by the implementation of DVFS method. An energy utilized for the data transmission from the data node to the computational node is interpreted as energy consumed by the data receive node (computational server) under the assumption that communication links between computing and data servers works in fully active mode during the data transmission and are hibernated into the sleep mode in the idle periods. This methodology is very similar to the general concept of the sleep link-based Adaptive Link Rate (ALR) on/off policy [4]. However, in our approach we consider just the logical network topology, where each computational node is connected to the data host by a distinct network link without additional transmission nodes.

The generic scheduling model presented in this paper is an extended version of our previously developed and published models [9], [10] for energy-aware scheduling in very simple grid architectures. This extension has been made by the aggregation of the data transmission and energy criteria with the conventional scheduling objective functions, and the development of simple genetic-based data- and energy-aware schedulers. These schedulers are integrated with *Green Data-Sim-G Batch* grid toolkit. Differently to the most of the approaches in batch scheduling, the required data is processed *during* the execution of the grid applications, not before the computation, and only small amount of data is delivered as a prior load to start the calculation process.

The remainder of this paper is structured as follows. First we define a modified Expected Time to Compute matrix model for data-aware independent batch scheduling and energy consumption model. A brief presentation of the genetic schedulers and main concept of *Green Data-Sim-G Batch* grid simulator is followed by a simple analysis of the experiments conducted for two variants of the genetic schedulers. The paper ends with simple conclusions and future research plan.

## II. Scheduling problem, model and objectives

### A. Notation

Let us introduce the following notation for the main system and scheduling components ( [8]): (i) $N_{batch} = \{t_1, \dots, t_n\}$ is a batch of $n$ tasks by the grid end-users; (ii) $M_{batch} = \{m_1, \dots, m_k\}$, is a set of $k$ computational resources (machines) available in the system; (iii) $F_{batch} = \{f_1, \dots, f_r\}$ is a set of $f$ data files needed for the completion of the tasks from $N_{batch}$; and (iv) $DH = \{dh_1, \dots, dh_s\}$ is a set of $s$ data hosts with the necessary data service capabilities.

The batch of tasks is characterized by a *batch workload vector* $WLoad_{batch} = [wload_1, \dots, wload_n]$, where $wload_j$ denotes a computational load of the task $t_j$ expressed in Millions of Instructions (MI). For meta-tasks (usually modelled by Directed Acyclic Graphs) this parameter must by estimated by using some local methods, in this paper *we do not* consider particular meta-task structures. It means that each meta-tasks is executed on single specified parallel machine or in single cluster. This is more realistic in our opinion than a scenario, in which the subtasks can be highly distributed among the global grid, and required data files are also distributed, which may increase significantly the time of finalizing the whole meta-task. Each task $t_j$ requires a set of files $F_j = \{f_{(1,j)}, \dots, f_{(r_j,j)}\}$ ($F_j \subseteq F_{batch}$) that are distributed on a subset $DH_j$ of the data nodes extracted from $DH$. We assume that each data host can serve multiple data files at a time and data replication is *a priori* defined as a separate replication process [8].

A *computing capacity vector* $CC = [cc_1, \dots, cc_k]$, ($cc_i$ denotes the computing capacity of the node $i$) expresses the ‘computational power’ of the whole system. Each $cc_i$ parameter ($i = 1, \dots, k$) can be expressed in MIPS (Million Instructions Per Second) calculated for CPUs, or in CPU clock frequencies. The prior loads of the computational nodes are defined by a *ready times vector* $ready\_times = [ready_1, \dots, ready_k]$. Gamma probability distribution is the most popular methodology of estimation of the workload and computing capacity parameters (see [7] , chapter 2 , for details).

### B. Estimation of task completion times

We estimate the times needed for the completion of the tasks submitted to the grid system with the transmission of the requested data files based on an *Expected Time to Compute (ETC)* matrix model [1]. We keep the original concept of *ETC* matrix $ETC = [ETC[j][i]]_{n\times k}$ ($ETC[j][i]$ denotes an expected (estimated) time needed for the computing the task $t_j$ at the resource $m_i$) for calculation of task execution times. In order to estimate the data transmission times, we defined a *Data Transmission Time* matrix $TT = [TT[j][i]]_{n\times m}$, where $TT[j][i][f_{(p,j)}]$ denotes a time needed for the transfer of the data file $f_{(p,j)}$ ($p \in \{1, \dots, r_j\}$) from the data host $dh_{(p,j)} \in D_j$ to the computational node $m_i$, and is calculated as follows [8]:

$$TT[j][i][f_{(p,j)}] = response_{time}(dh_{(p,j)}) + \frac{Size\left[f_{(p,j)}\right]}{B(dh_{(p,j)}, i)}. \qquad (1)$$

We denoted by $response_{time}(dh_{(p,j)})$ in Eq. 1 the time needed for transfering the data file $f_{(p,j)}$ to the computational node $m_i$ from the data host $dh_{(p,j)}$, and by $B(dh_{(p,j)}, i)$ – a bandwidth of the (logical) link between $dh_{(p,j)}$ and $m_i$. In the presented model we abstract the physical network between the data hosts and computational and restrict our analysis just to a logical network topology where each computational node is connected to the data host by a distinct network link.

The estimated completion time for a given task on a specified machine can be defined as the wall-clock time taken for the task from its submission till completion and is a function of computing and transmission times specified in Eq. (1). The impact of the data transfer time on the task completion time depends on the mode, in which the data files are processed by the task. There are two major scenarios of such data processing: (i) the data files needed for the task execution are transferred simultaneously *before* the task execution; and (ii) some of the data files are transferred completely prior to the task execution. We focus in this paper on the second scenario (See [8], Sec. 3.2 and Fig. 4b).

### *C. Energy Model*

The energy model for data-aware scheduling is mainly based on DVFS technique [2]. In this paper, we restrict our analysis just to the optimization of energy consumed during the computation of the tasks and receiving the data from the data hosts. It is assumed that the communication links between computational and data nodes during the data transfer are fully active (and open) and are switched into the sleep mode after finishing the data download. We assume that the communication links between data and computational servers are direct without any transitive nodes. So no extra transition delays as well as request sending times are considered. Obviously, this model cannot be directly implemented in the realistic grid, but it was successfully implemented in backbone network-based middle-size clusters.

DVFS model implemented in CPM module is based on the power consumption model employed in complementary metal-oxide semiconductor (CMOS) logic circuits [12]. In this model, the capacitive power $Pow_{ji} = A \cdot C \cdot v^2 \cdot f$ utilized by the machine $m_i$ for computing the task $t_j$ depends on the number of switches per clock cycle voltage $A$, total capacitance load $C$, supply voltage $v$ and machine frequency $f$. The reduction of the supply voltage and frequency is directly correlated to the reduction of the energy utilization. Table I shows the parameters for 16 DVFS levels and three main 'power supply' categories for machines defined for the grid system employed in this study.

TABLE I
DVFS LEVELS FOR THREE MACHINE CLASSES

| | Class I | | Class II | | Class III | |
|---|---|---|---|---|---|---|
| **Level** | **Volt.** | **Rel.Freq.** | **Volt.** | **Rel.Freq.** | **Volt.** | **Rel.Freq.** |
| 0 | 1.5 | 1.0 | 2.2 | 1.0 | 1.75 | 1.0 |
| 1 | 1.4 | 0.9 | 1.9 | 0.85 | 1.4 | 0.8 |
| 2 | 1.3 | 0.8 | 1.6 | 0.65 | 1.2 | 0.6 |
| 3 | 1.2 | 0.7 | 1.3 | 0.50 | 1.9 | 0.4 |
| 4 | 1.1 | 0.6 | 1.0 | 0.35 | | |
| 5 | 1.0 | 0.5 | | | | |
| 6 | 0.9 | 0.4 | | | | |

The reduction of the machine frequency and its supply voltage can lead to the extension of the task execution and completion times as well as for the extension of times needed for downloading the data from the data servers. For a given 'task-machine' pair $(t_j, m_i)$, the energy consumed for the completion of the task $t_j$ on machine $m_i$ at the DVFS level $s_i$ can be calculated as follows:

$$E_{ji}(s_i) = \gamma \cdot (f_{s_i})_j \cdot f \cdot [(v_{s_i})_j]^2 \cdot completion[j][i][s_i], \qquad (2)$$

where $\gamma = A \cdot C$ is a constant parameter for a given machine's class, $(v_{s_i})_j$ is a voltage supply for machine $m_i$ at level $s_i$ for computing task $t_j$, and $(f_{s_i})_j$ is a corresponding relative frequency for machine $i$. The completion time $completion[j][i][s_i]$ is calculated as in the following way:

$$\begin{aligned} & completion[j][i][s_i] = \max\nolimits_{f_{(p,j)} \in \widehat{F}_j} TT[j][i][f_{(p,j)}] \\ & + \left[ \sum\nolimits_{f_{(l,j)} \in \left[F_j \setminus \widehat{F}_j\right]} TT[i][j][f_{(l,j)}] \right] \\ & \cdot \frac{ETC[j][i]}{f_{s_i})_j} \end{aligned} \qquad (3)$$

The cumulative energy utilized by the machine $m_i$ working at the power supply level $s_i$ for the completion of all tasks from the batch and downloading of all requested data files this machine, is defined in the following way:

$$\begin{aligned} E_i(s_i) = \quad & \sum\nolimits_{\substack{j \in Tasks(i) \\ l \in \hat{L}_j}} E_{ji}(s_i) + \gamma \cdot f [v_{s_{max_i}}]^2 \cdot ready_i + \\ & \gamma \cdot f_{s_{min_i}} \cdot f \cdot \cdot [v_{s_{min_i}}]^2 \cdot Idle[i][s_i] \end{aligned} \qquad (4)$$

where $Tasks(i)$ is a set of tasks assigned to machine $m_i$, $ready_i$ is the ready time of machine $m_i$, $Idle[i][s_i]$ denotes an idle time of $m_i$, $\hat{L}_i$ denotes a subset of DVFS levels used for the tasks assigned to $m_i$, and $s_{min_i}$ and $s_{max_i}$ denote the minimal and maximal power supply levels specified for $m_i$.

An average cumulative energy utilized by the grid system for completion of all tasks in the batch is defined as follows:

$$E_{batch} = \frac{\sum_{i=1}^{m} E_i(s_i)}{m} \qquad (5)$$

### *D. Scheduling Scenarios and Objectives*

Similar to our previous works (see e.g. [7]), we consider two scheduling scenarios based on the power supply models:

1) **I – Max-Min Mode**, where all machines work with maximal power supply when a computation is provided and data is received from the data node, and are switched into the sleep mode in the computational idle periods;
2) **II – Modular Power Supply Mode**, where each machine can work at **different** DVFS levels during the calculations.

In each of those scenarios, we define the scheduling problem as a bi-objective global optimization task, where two major scheduling criteria, namely (i) makespan and (ii) cumulative energy utilized by the system, are minimized. This minimization is realized as two-step hierarchical procedure, in which makespan is defined as a dominant criterion. Based on the general ETC matrix model, makespan can be defined in terms of the completion times of the machines as the finishing time for the last task in the batch, which is specified as the maximal completion time of all machines available in that batch, that is to say:

$$makespan = \max_{i=1}^{m} completion[i][s_i]. \qquad (6)$$

where $completion[i][s_i]$ is the completion time of the machine $m_i$ working at the level $s_i$, which is defined as follows:

$$completion[i][s_i] = ready_i + \sum_{j \in Tasks(i)} \frac{1}{f_{s_i}} \cdot completion[j][i][s_i] \quad (7)$$

The second step of the scheduling optimization procedure is the minimization of the total energy consumed in the grid for scheduling a given batch of tasks. In **Modular Power Supply** mode, the average cumulative energy $E_{batch}$ is given by Eq. (5). In **Min-Max** mode, the average cumulative energy is denoted by $E_I$ and can be calculated by using Eq. (5), in which $s_i = s_{max_i}$.

In both cases $E_I$ and $E_{batch}$ are minimized and subject to the following constraint:

$$\sum_{l \in \hat{L}_i} \left[ \frac{1}{f_{s_i}} \cdot completion[j][i][s_i] \right] \leq \ makespan; \ \forall i \in \{1, \ldots, m\}, \quad (8)$$

where $\hat{L}_i$ denotes a subset of DVFS levels specified for tasks assigned to machine $m_i$.

## III. Empirical analysis

In this section we present the results of a simple empirical analysis of the performance of two implementations of GA-based energy and data-aware schedulers for static and dynamic versions of the independent batch scheduling problem in grid. We have developed a *Green Data-Sim-G Batch* simulator, which is a modified version of our previous *Green Sim-G Batch* grid simulation toolkit (see [7]) by the implementation of a data processing module as it is presented in Fig. 3.

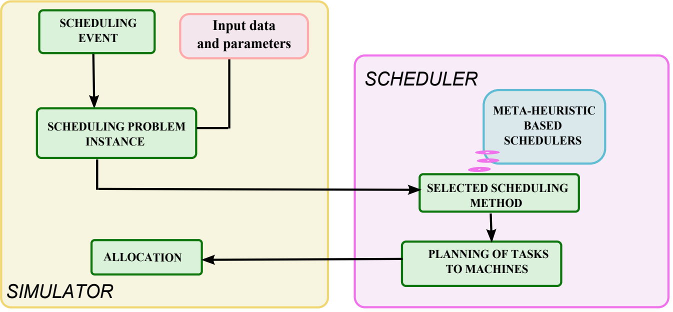
SCHEDULING EVENT
Input data and parameters
SCHEDULING PROBLEM INSTANCE
ALLOCATION
SIMULATOR
SCHEDULER
META-HEURISTIC BASED SCHEDULERS
SELECTED SCHEDULING METHOD
PLANNING OF TASKS TO MACHINES

Fig. 3. General concept of *Green Data Sim-G Batch*

This data processing module is responsible for generating (i) a set of data files, (ii) a set of data hosts, (iii) data transmission time matrix, (iv) response time vector, and (v) bandwidth vector. All those data are considered as basic characteristics of an instance of the problem and together with (vi) workload vector of tasks, (vii) computing capacity vector, (viii) prior load vector, and (ix) ETC matrix are passed on to the selected scheduler, which computes the schedule of the task assignments to the machines. Finally, the scheduler sends the schedules to the simulator, which makes the allocation.

### A. Genetic-based schedulers

For solving the batch scheduling problem addressed in this paper, we used two implementations of simple genetic grid schedulers, namely *Plain Genetic Algorithm Scheduler* (*GA*) and *Struggle Genetic Algorithm Scheduler* (*StGA*), both with different replacement mechanisms. Genetic-based schedulers can easily explore the robustness of the search space and they can tackle various scheduling attributes [14]. We have already used such methods in our previous works, for the big set of benchmarks and instances of the problem has been defined [7]. The general frameworks of those schedulers are based on classical $(\mu + \lambda)$ evolutionary strategy (see e.g. [13]), adapted to the scheduling problem through the implementation of the following genetic operators:

- **Initialization method:** Randomly generated initial population;
- **Selection method:** Linear Ranking Selection;
- **Crossover operator:** Partially Mapped Crossover (PMX);
- **Mutation operator:** Move;
- **Replacement operators:** Elitist Generational (GA) and Struggle (StGA).

The chromosomes in the populations are represented as vectors $S = [i_1, \ldots, i_n]^T$ of machine labels ($i_j \in M_l$ is a label of the machine in which the task $j$ is executed). All the above genetic operators are popular in grid scheduling and their detailed definitions can be found in [7].

### B. Input data and key parameters for simulator and schedulers

The performance of genetic-based schedulers analyzed in two types of grid environment: (i) static and (ii) dynamic, and four grid size scenarios: (i) small grid (32 hosts/512 tasks), (ii) medium grid (64 hosts/1024 tasks), (iii) large grid (128 hosts/2048 tasks), and (iv) very large grid (256 hosts/4096 tasks). The schedulers' key parameters, including mutation and crossover probabilities, population size and stopping criteria (can be the maximal number of evolution steps or termination time criterion, are presented in Table II.

TABLE II
Shedulers' key parameters for static and dynamic benchmarks.

| Parameter | GA | StGA |
|---|---|---|
| **evolution steps** | $20 * m$ | |
| **pop. size** ($pop\_size$) | $4 * (\log_2(m) - 1)$ | |
| **cross probab.** | 0.9 | 1.0 |
| **mutation probab.** | 0.2 | |
| **termin. time crit.** | 40 secs (*static*) / 75 secs (*dynamic*) | |

The values of key parameters for the *Green Data Sim-G Batch* simulator for static and dynamic grid scenarios are presented in Table III ($N(*, **)$ denotes the Gaussian distribution), and are similar to those used for *Green Sim-G Batch* in our previous work (see [7] for the detailed interpretation of all parameters, wide list of references and comprehensive empirical analysis results for *Green Sim-G Batch*).

The performance of two genetic schedulers in two scheduling scenarios was measured through the makespan criterion and a relative energy consumption improvement rate expressed as follows:

$$Im = \frac{E_I - E_{batch}}{E_{batch}} \cdot 100\%, \quad (9)$$

TABLE III
Parameter setting for the grid simulator static instances

| | Small | Medium | Large | Very Large |
|---|---|---|---|---|
| Static Instances | | | | |
| *Resource cap. (in MIPS)* | $N(1000, 175)$ | | | |
| *Workload of tasks* | $N(25 \cdot 10^7, 4375 \cdot 10^4)$ | | | |
| Dynamic Instances | | | | |
| *Total hosts* | 32 | 64 | 128 | 256 |
| *Init. hosts* | 27 | 58 | 121 | 248 |
| *Add host* | $N(625000, 93750)$ | $N(562500, 84375)$ | $N(500000, 75000)$ | $N(437500, 65625)$ |
| *Delete host* | $N(625000, 93750)$ | | | |
| *Total tasks* | 512 | 1024 | 2048 | 4096 |
| *Init. tasks* | 384 | 768 | 1536 | 3072 |
| *Workload* | $N(25 \cdot 10^7, 4375 \cdot 10^4)$ | | | |

where $E_{batch}$ and $E_I$ are defined as in Sec. II-D.

Each experiment was repeated 30 times under the same configuration of operators and parameters.

### C. Results

We present in Fig. 4 and Fig. 5 the box-plots of the makespan values (confidence level - 95 %) achieved in 30 runs of the experiments in **Min-Max** and **Modular Power Supply** modes. The makespan is measured and expressed in arbitrary time units specified for the simulator input data.

It can be observed that the implementation of struggle replacement mechanism in the genetic scheme has a significant impact on the minimization of the makespan values. The *StGA* algorithm achieved better results than *GA* scheduler in all but Large static and Small dynamic grids in **Min-Max** scheduling scenario and all grid configurations in **Modular Power Supply** scenario. Similar results has been achieved in our previous work, when data transfer times have been ignored (see [7], Chapter 4 for details). However, compare to those results, the values of makespan achieved in the experiments provided in this study have increased average by 10–25 %, which confirms the importance of data transfer and access criteria in the analysis of the whole scheduling process. The ranges in the achieved makespan values for both genetic meta-heuristics are not greater than 30-40 % of the mean makespan values, which means that the stability of the schedulers in all cases are acceptable. The distributions of the makespan results are asymmetric: the skewness in the static case is positive in most of the grid configurations and both scheduling scenarios, and it is negative in the dynamic grids in all problem instances. It means that the reduction of the average makespan in this case is much harder than in static case (the mean values are closer the third quantile, than the first one), which confirms the complexity of the problem in the realistic dynamic grid scenarios.

The box-plots for the energy saving rates $Im(E)$ are presented in Fig. 6 and Fig. 7.

The results of the energy optimization are different compare with the makespan results. In this case the struggle scheduler outperforms the *GA* algorithm just in the dynamic instances. It means that in static case it works quite good in **Min-Max** scenario, so no additional DSV modules are necessary here. The range of the average saving rate values is 16%–50% for most of the considered scheduling and grid scenarios, which is rather high, and higher than in similar test provided without data processing module activated in the simulator, where it was 10-35% [10]. It means that in the case of data-aware scheduling the scaling of the power supply of the network computational nodes may play a crucial role in the energy saving. The skewness of the distribution of the results is positive in most of the static instances and negative in all dynamic ones.

## IV. Conclusions and future work

In this paper we have addressed a general problem of the optimization of cumulative energy consumption in data-aware genetic-based scheduling problem of tasks submitted independently by the grid end-users. The energy management model is based on *DVFS* technique adapted to the dynamic grid environment. We formalized the scheduling problem as a bi-objective optimization task with makespan as a dominant criterion.

Certainly, simulation is just the first step in comprehensive empirical study of the problem and developed schedulers' performance. However, our model can be easily adapted to various realistic scheduling scenarios and realistic grid infrastructures, such as the large-scale banking system or highly distributed data system. First, we do not consider any special architectures for grid resources, which means that this characteristics can be specify separately and integrated with the system simulator. The term "task" can be also used for monolithic applications, metatasks or parallel applications represented by *Directed Acyclic Graphs*. The schedulers are integrated with the main grid simulator as separate modules, and therefore they can be easily modified, extended and hybridized with the other algorithms. Finally, we simulate the dynamics of the realistic grid system, in which the availability of the resources and the number of tasks may vary over the time. The implementation and validation of the presented model for the realistic grid testbeds, its application in cloud and mobile cloud environments and a simple formal analysis of this model are the main targets of our further research in data and energy-aware scheduling.

## References

[1] Ali, S., Siegel, H.J., Maheswaran, M., and Hensgen, D.: Task execution time modeling for heterogeneous computing systems, *Proceedings of Heterogeneous Computing Workshop*, pp. 185–199, 2000.

[2] Babaii-Rizvandi, N., Taheri, J., Zomaya, A.Y.: Some Observations on Optimal Frequency Selection in DVFS-based Energy Consumption Minimization, *Journal of Parallel and Distributed Computing*, Vol. 71(8), pp. 1154-1164, 2011.

[3] Garg, S.K., Yeo, C.S., Anandasivam A., and Buyya R.: Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers, *J. of Parallel Distrib. Comput.*, Vol. 71(6), pp. 732–749, 2011.

[4] Gunaratne, C., Christensen, K., Nordman, B. and Suen, S.: Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR), *IEEE Transactions on Computers*, Vol. 57, pp. 448-461, 2008.

[5] Kliazovich, D., Bouvry, P., and Khan, S.U.: DENS: Data Center Energy-Efficient Network-Aware Scheduling, in *Proc. of the ACM/IEEE Int. Conference on Green Computing and Communications (GreenCom), Hangzhou, China*, pp. 69–75, 2010.

[6] Khan, S. U., and Ardil, C.: "A Weighted Sum Technique for the Joint Optimization of Performance and Power Consumption in Data Centers", *International Journal of Electrical, Computer, and Systems Engineering*, Vol. 3, No. 1, pp. 35–40, 2009.

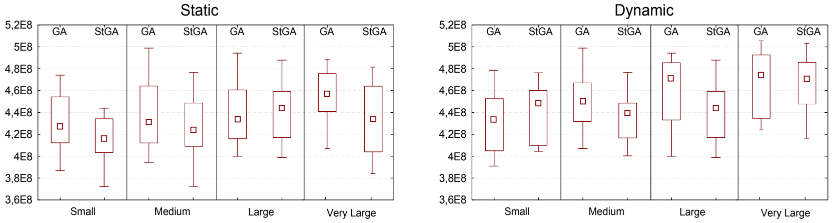

Fig. 4. The box-plot of the results for makespan in **Min-Max** scheduling scenario.

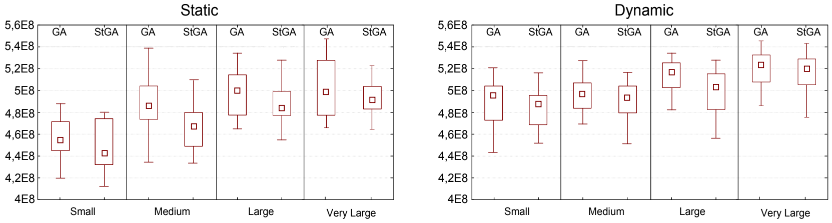

Fig. 5. The box-plot of the results for makespan in **Modular Power Supply** scheduling scenario.

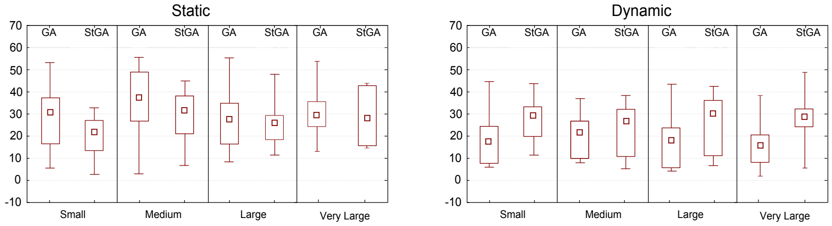

Fig. 6. The box-plot of the results for relative energy saving rates (in %) in **Min-Max** scheduling scenario.

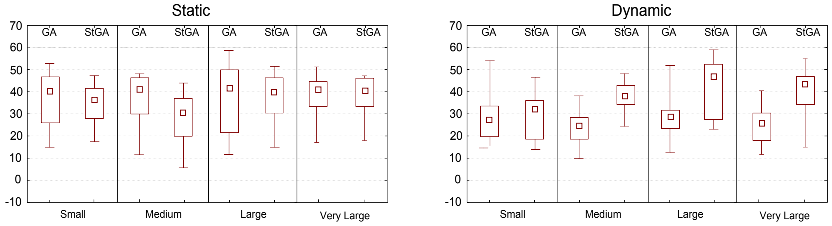

Fig. 7. The box-plot of the results for relative energy saving rates (in %) in **Modular Power Supply** scheduling scenario.

[7] Kołodziej, J.: *Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems*, in *Studies in Computational Intelligence* Springer series, Vol. 419, Springer Vlg., Berlin-Heidelberg, 2012.

[8] Kołodziej, J. and Khan, S. U.: Data Scheduling in Data Grids and Data Centers: A Short Taxonomy of Problems and Intelligent Resolution Techniques, *Transactions on CCI*, Vol X, LNCS 7776, pp. 104–121, 2013.

[9] Kołodziej, J., Khan, S.U., and Xhafa, F.: Genetic Algorithms for Energy-aware Scheduling in Computational Grids", in Proc. of the *6th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC2011)*, 2011.

[10] Kołodziej J. , Khan S.U., Wang L., Byrski A., Min-Allah N. and Madani S. A.: Hierarchical Genetic-based Grid Scheduling with Energy Optimization, *Cluster Computing*, DOI: 10.1007/s10586-012-0226-7, 2013.

[11] Kosar, T., and Balman, M.: A new paradigm: Data-aware scheduling in grid computing, *Future Gener. Comput. Syst.*, Vol. 25(4), pp. 406–413, 2009.

[12] Lorch, J.R., and Smith, A.J.: Improving dynamic voltage scaling algorithms with pace, In *2001 ACM SIGMETRICS international conference on measurement and modeling of computer systems*, pp. 50-61, 2001.

[13] Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 1992.

[14] Venugopal, S., and Buyya, R.: An SCP-based heuristic approach for scheduling distributed data-intensive applications on global grid, *J. Parallel Distrib. Comput.*, Vol. 68, pp. 471–487, 2008.

[15] Wang, L., Khan, S.U., Chen, D., Kolodziej, J., Ranjan, R., Xu, C.-Z., Zomaya A.Y.: Energy-aware parallel task scheduling in a cluster, *Future Generation Comp. Syst.*, Vol. 29(7), pp. 1661–1670, 2013.