ITC 1/47
Journal of Information Technology and Control
Vol. 47 / No. 1 / 2018
pp. 93-106
DOI 10.5755/j01.itc.47.1.18062
© Kaunas University of Technology

A Comparison of DMS Workflows Scheduling Concepts

Received 2017/04/26 | Accepted after revision 2018/01/08

http://dx.doi.org/10.5755/j01.itc.47.1.18062

A Comparison of DMS Workflows Scheduling Concepts

Nemanja Nedić
Schneider Electric DMS Novi Sad, Novi Sad, Serbia, e-mail: nemanja.nedic@schneider-electric-dms.com

Goran Švenda
Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia, e-mail: svenda@uns.ac.rs

Corresponding author: nemanja.nedic@schneider-electric-dms.com

The development of computational and communication technology enabled the introduction of a Distribution Management System (DMS) as a sophisticated approach to solve various problems regarding the power supply in modern society. These advanced systems execute a large number of workflows to meet the requirements and provide the desirable functionalities. Very high demands for computational resources, during the DMS operation, imposed a necessity for an adequate DMS workflow management. Three DMS workflow scheduling concepts (architectures) are presented in this paper: centralized, mixed and hierarchical. These scheduling concepts are compared by different parameters, such as performance, scalability and complexity of implementation. The conducted experimental study shows that the hierarchical scheduling concept not only provides the best performance, but is also the most scalable and entirely eligible in terms of implementation complexity.

KEYWORDS: Workflow Scheduling, Workflow Management System, Grid Computing, Distribution Management System.

Abbreviations:

CB	–	Computing Broker	DN	–	Distribution Network
CPU	–	Central Processing Unit	GDS	–	Graphical Data Service
CS	–	Computing Service	GIS	–	Geographical Information Service
DAG	–	Direct Acyclic Graph	SCADA	–	Supervisory Control and Data Acquisition
DDS	–	Dynamic Data Service	SDS	–	Static Data Service
DMS	–	Distribution Management System	WMS	–	Workflow Management System

1. Introduction

1.1. Motivation

Distribution Management System (DMS) is developed as an industry-grade product which manages the entire distribution network (DN) [9]. Nowadays, distribution systems are vast. Therefore, a specific data model (network model) [7], used to represent a DN, may describe several tens of millions of entities [6]. Additionally, the complexity of DMS is reflected by an extensive number of main requirements that have to be met: control and monitoring, smart operation, DN analysis, optimization and planning, demand-response management and management of outages. To meet these expectations, DMS is tightly integrated with other systems like Supervisory Control and Data Acquisition, Customer Information System, Meter Data Management, Weather Information System, Asset Management and Geographical Information System.

Such DMS, integrated into the Smart Grid concept [3, 29], uses a Grid [2, 15, 20] to store and process aforementioned large amounts of data so that required functionalities are provided [25]. Each functionality demands execution of different (most commonly dependent) *tasks* on Grid resources. A set of tasks whose execution completes a certain functionality determines a *workflow* [14].

DMS operates as a reactive system [17, 27]. This category of computer system is characterized by an interaction with computational environment during which information exchanges occur. Their reaction to the influences coming from the environment is an initiation of different activities (workflows) which can result in a state modification. DMS receives various stimulations for generating workflows: actions generated via a user interface, changes of topology and measured values from a SCADA [3] or model changes from a GIS [38]. The interaction can happen in any random moment, so the full set of workflows that has to be executed (*workflow application*) is not known in advance, it evolves in time instead. Furthermore, workflows may be mutually dependent [25] and prioritized [21] (e.g. processing of an alarm event is more important than execution of "what-if" analyses [25]). Consequently, the additional complexities are included in the workflow management.

The Grid workflow management system provides scheduling process to coordinate the execution of workflows in order to assign a stream of tasks to available resources with a goal to optimize various performance metrics [13, 22]: minimization of single workflow execution time, fairness of load distribution or minimization of *makespan* (execution time of workflow application) [8, 19].

Behavior of Grid resources is not predictable in sufficiently good measure [41]. For example, during workflow execution, some resources suddenly may get busy with internal tasks, while a computer network can be overloaded for some time period. For these reasons, the estimated time when a task will be executed often does not coincide with the time when the task is really completed. Thus, unsteadiness of computational resources is another difficulty which a workflow management has to handle.

Based on the previous discussion, the scheduling process has to deal with the following challenges:

_ manipulation with large data workflows;

_ examination of the dependency of workflows;

_ consideration of priority of workflows;

_ dynamic environment – unpredictable behavior of Grid resources and dynamical arrival of workflows to execution;

_ optimal utilization of Grid resources.

1.2. Literature Review

Two types of scheduling algorithms are thoroughly researched, namely, static [4, 32], which can be used only when all workflows are known prior, and dynamic [23, 32], which must be utilized when circumstances are unforeseeable. Dynamic algorithms are necessary for adequate DMS operation, because workflows are generated at random timestamps and fluctuation of computing performance is constantly present [41]. Some papers propose solving similar problems by using artificial intelligence, like neural networks [39] and genetic algorithms [26]. On the other hand, some approaches are based on collecting information about Grid, executing predefined algorithms and making scheduling decisions [10, 11, 14].

The chosen concept (architecture) of workflow scheduling may have a significant impact on the system operation [16]. Different concepts may use different scheduling algorithms to provide parallel and distributed task execution, better utilization of computational resources which results in overall increased performance – the speed of workflow execution.

The centralized scheduling concept implies existence of one scheduling unit which coordinates the execution of all workflows [16]. Opposite of the centralized concept, the distributed scheduling includes multiple units which make decisions about workflow management [14]. Therefore, one unit usually holds information about a subset of workflows assigned to it (a part of workflow application). If work of distributed scheduling units is not coordinated [31], the best decisions about a subset of workflows made by one scheduling unit may not ensure the best performance of the whole workflow application [24]. Hierarchical scheduling is conducted with one central scheduling unit and some low-level scheduling units [5]. The central unit does the preliminary examination of workflow application and sends workflows to low-level units which are responsible for assigning common tasks to Grid resources.

1.3. Aim and Contribution

This paper aims to compare qualities of different workflow scheduling concepts, like performance, scalability and complexity of implementation. Each concept has to consider requirements listed in the *Motivation section*.

Makespan is used to quantify performance which is the main indicator of feasibility and successfulness of a scheduling concept [41]. For these purposes, practical experiments, which simulate the DMS real-time control, are developed.

By analyzing design, we will present the complexity of implementation and explain scalability of each workflow scheduling concept.

1.4. The Organization of the Paper

After Introduction, the general architecture of a DMS and its main workflows are presented in the second section. Section 3 explains the model of the workflow application, which is the starting point of the efficient operation of the Workflow Management System that is introduced in Section 4. Section 5 presents an experimental study, while Section 6 contains final remarks and conclusion.

2. DMS Grid Architecture and Workflows

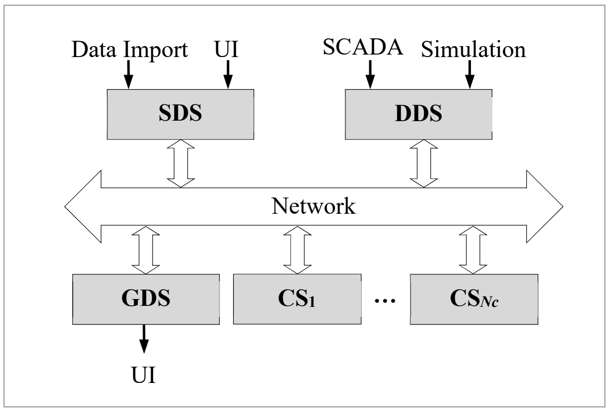
Diverse types of data are processed during DMS operation [25]:

- static data (rarely changed after the initial import) fully describe a DN, entities and their connectivity;
- dynamic data (frequently changed) represent dynamic states of the devices within a DN, e.g. voltage values;
- graphic data are used for visualization of a DN.

DMS is built as a collection of services adapted to operate in a distributed computing environment. In order for DMS to operate correctly, it is essential that services cooperate and exchange information. Each service implements a specific functionality while preserving data to provide them an effective, secure and continuous operation. Figure 1 presents the core services of DMS [25]:

- **Static Data Service** (SDS) provides read/write actions on the static data. Most commonly, SDS relies on a relational database which is hosted on the current computational resource.
- **Dynamic Data Service** (DDS) caches the dynamic data. Since these data change frequently, huge time-series data about process variables are stored.
- **Graphical Data Service** (GDS) is responsible for maintaining the graphical data. It enables each user client application only to maintain its current view of the DN in its memory.
- **Computing Services** (CSs) are numerous since they provide means for execution of DMS functions [35]. DMS functions are fairly burdensome and the most frequent workflows [25]. They receive static and correspondent dynamic data for a part of the DN, and perform a requested set of calculations. In respect to the requirements of complex calculations, CSs are deployed on resources with powerful CPUs. Tasks executed on this type of DMS service are called *computing tasks*. Figure 1 shows a Grid with N_c computing services (CSs).

Figure 1
DMS services deployed on a computational Grid

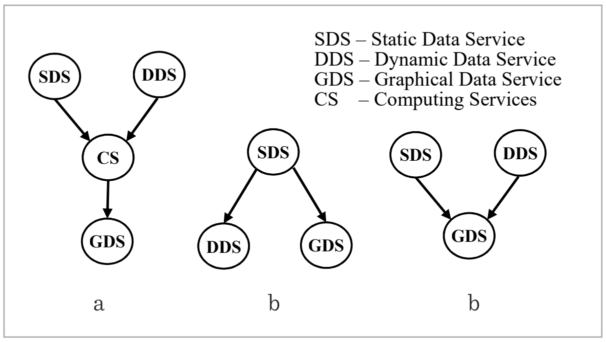


The Grid uses a peer-to-peer communication model which is convenient for large data transfers [30]. It provides the least time losses while transferring data between computational resources [33].

As a result of analyzing requirements and functionalities, the most common DMS workflows are the following:

- **DMS Function Execution** is triggered when a change of the state of a device occurs. It includes four tasks – a part of dynamic data and correspondent static data are transferred to a CS, so that relevant calculations may be done. Consequently, graphical data are updated with results (Figure 2(a)).
- **Model Update** is a workflow which is executed when static data have to be updated. It consists of three tasks – based on the modification of static data, it is necessary to update graphical and dynamic data (Figure 2(b)).

Figure 2
DMS workflows: a) DMS Function Execution, b) Model Update, c) Graphical View Refresh


SDS – Static Data Service
DDS – Dynamic Data Service
GDS – Graphical Data Service
CS – Computing Services
SDS
DDS
CS
GDS
a
SDS
DDS
GDS
b
SDS
DDS
GDS
b

- **Graphical View Refresh** is initiated as a possible GDS error recovery. Three tasks are executed during this workflow – a part of static data and correspondent dynamic data are transferred to GDS, so that graphical data may be refreshed (Figure 2(c)).

Each DMS workflow can be represented as a set of tasks with dependencies between them [37] by a directed acyclic graph (DAG) [28]. The vertices of the DAG represent tasks, while the edges are data dependencies between them. A task becomes executable when all its dependencies are met: all predecessor tasks are completed and input data required by the task are available at a Grid resource.

Workflows which arrive for execution are ranked by the priority. All tasks included in a workflow take over the same priority as the workflow. Furthermore, dependencies between workflows may occur, e.g. execution of DMS functions over a part of a DN is directly dependent on the initial import of that DN part [25].

Since historical data can be used to evaluate the workload of tasks [18], a task T_i is characterized with the following properties:

p_i – priority;

t_i – estimated time to execute the task (which corresponds to the estimated task workload w_i);

n_i – execution resource (DMS service, $n_i \in \{SDS, GDS, DDS, CS\}$);

P_{T_i} – set of predecessor tasks;

S_{T_i} – set of successor tasks.

Scheduling algorithms use these properties to choose the optimal allocation of resources for task execution. Optimization goal is to rearrange incoming workflow tasks in order to support the priority and get maximum usage of all Grid resources, so that the makespan is minimized.

3. The Workflow Application

This section provides a description of the workflow application model [25] which is further used to define optimization problem.

3.1. The Workflow Application Model

Workflow application is built by connecting tasks of dependent workflows. Output tasks of a "parent" workflow become predecessors for all input tasks of a "child" workflow, therefore, the workflow application is also represented as a DAG. Tasks without any predecessors are input tasks, while tasks without any successors are output tasks of a workflow application.

An efficient workflow scheduling requires quality information about the managed workflow application. For that reason, the workflow application model is enriched with additional data. Every node and edge of the workflow application DAG receive a weight. The weight of a node is the estimated execution time of a task, while the weight of an edge represents the time necessary to transfer data between computational Grid resources on which the task and its successor are executed.

Data transfer rates (the amount of data which can be transferred via network) between computational resources are well-known. The transfer rate between resources n_i and n_j is denoted by $rate_{n_i n_j}$. The amount of data that needs to be transferred after the completion of task T_i and before the start of task T_j, is denoted by $data_{T_i T_j}$. The communication time cost of edge (*i, j*) is defined as:

$$c_{T_i T_j} = \frac{data_{T_i T_j}}{rate_{n_i n_j}}. \tag{1}$$

IF st_{T_i} and ct_{T_i} are defined as the start time and the completion time of a task T_i, respectively, and nt_{n_i} as the earliest available time of a resource n_i, then the task T_i becomes eligible for execution at the timestamp

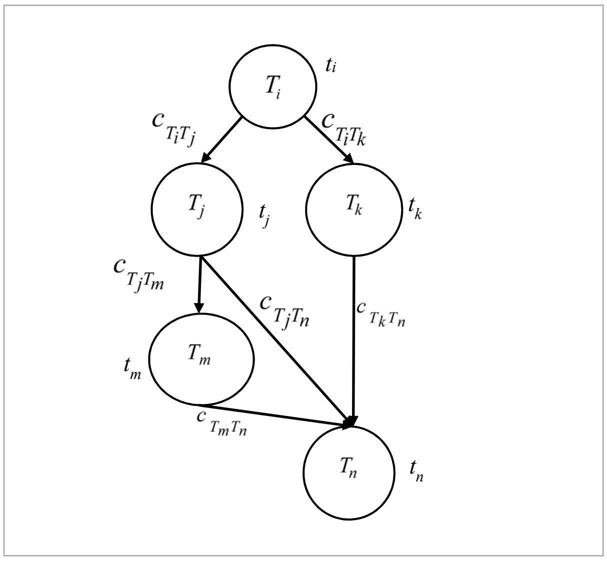
$$st_{T_i} = \max\left\{ \max_{T_j \in P_{T_i}} \{ct_{T_j} + c_{T_i T_j}\},\ nt_{n_i} \right\}. \tag{2}$$

he completion time ct_{T_i} is calculated as the sum of the start time st_{T_i} and the estimated execution time t_i:

$$ct_{T_i} = st_{T_i} + t_i. \tag{3}$$

Finally, the model of workflow application acquires its definite form which fully describes all workflows, task dependences and their properties (see Figure 3).

Figure 3
The workflow application model



3.2. The Optimization Problem

The makespan represents the objective function of the addressed optimization problem. Information held by the workflow application model is sufficient to define it [8]:

$$makespan = \max_{T_i \in Outputs} \{ct_{T_i}\}. \tag{4}$$

Outputs is the set of output tasks.

The optimization goal is to assign tasks to resources in order to minimize the makespan, respecting the actual constraints: dynamic environment, task dependencies and priorities.

4. The Workflow Management System

A Workflow Management System (WMS) is the backbone of the workflow manipulation within a Grid. It does not execute tasks, but coordinates them and ensures that tasks are carried out in a proper order and that their execution is done simultaneously whenever is possible [13, 22]. A WMS may support different scheduling concepts, like centralized [16], hierarchical [31], distributed [5], or some derived implementations [14, 25].

4.1. The Centralized Scheduling Concept

The basic WMS developed for centralized scheduling concept is presented in Figure 5. In the following sections, while elaborating mixed and hierarchical scheduling concepts, basic modifications related to Figure 5 will be pointed out.

WMS (see Figure 5) consists of two major components, *DAG Manager* and *Task Scheduler*. Moreover, it implements a scheduling algorithm to provide support for the high-throughput scheduling process. The algorithm searches for executable tasks whose completion will have the greatest impact on the overall performance – makespan. To ensure this, during the execution of the scheduling algorithm an *algorithmic parameter* whose value corresponds to the influence of a task on the makespan is calculated for every task. Therefore, the value of the algorithmic parameter determines which *ready task* (a task without predecessors) has an advantage during the allocation of Grid resources. It should be noticed that ready tasks are mutually independent since they have no predecessors.

4.1.1. The Algorithmic Parameter

The algorithmic parameter is determined based on the *Critical Path Algorithm* [34, 36]. This approach aims to determine the longest of all execution paths for every task [25]. It answers the question on *which tasks have the most impact on the overall execution time*.

In accordance with this, the tasks are ordered decreasingly by their rank (the algorithmic parameter). The rank of each output task corresponds to its execution time:

$$\underset{T_i \in Outputs}{rank_{T_i}} = t_i \,. \tag{5}$$

The ranks of other tasks are calculated recursively:

$$rank_{T_i} = t_i + \max_{T_j \in S_{T_i}} \{c_{T_i T_j} + rank_{T_j}\}\,. \tag{6}$$

4.1.2. The DAG Manager

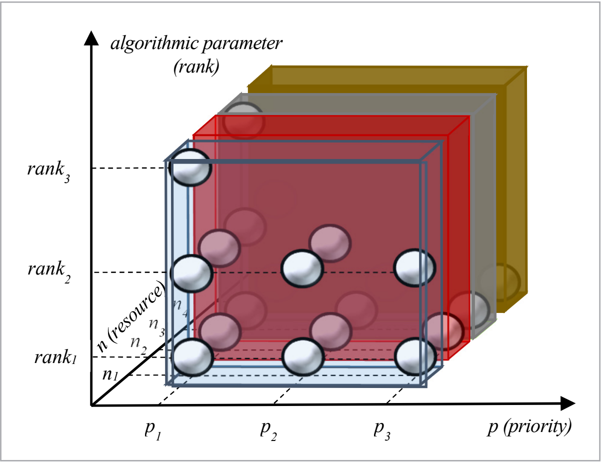
The workflow application is represented via a DAG and it is continually evolving during the system operation. The DAG Manager manipulates the workflow application and its main mission is to provide sorted ready tasks (according to algorithmic parameter and priority, at the time when a resource is available). The priority has greater influence on assigning task to resources than algorithmic parameter. If one task has higher priority and lesser value of algorithmic parameter than another task, the first task will acquire an advantage while mapping tasks to resources.

The DAG Manager monitors for a new workflow arrival or a task completion. When any of the events occur, the DAG is updated: a new workflow is included, while the completed task is removed from the workflow application.

As a consequence, the dependency information is updated and an analysis of DAG is initiated. The result of the analysis is a collection of ready and independent tasks. Subsequently, the algorithmic parameter is calculated. The collection of ready tasks is then classified according to priority, execution resource and algorithmic parameter (Figure 4).

Figure 4

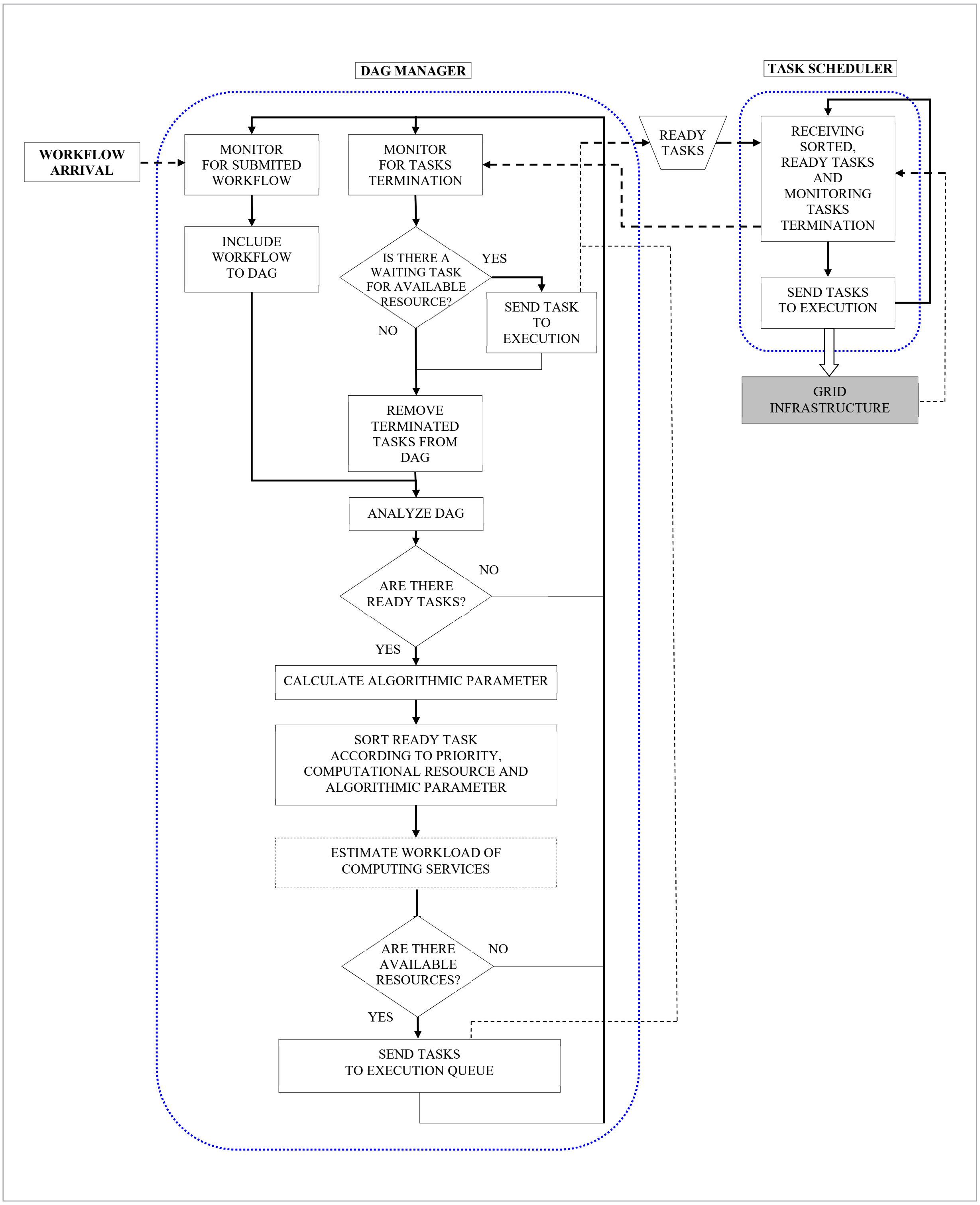
Ready tasks sorted according to priority, execution resource and algorithmic parameter



The basic idea is that when a Grid resource becomes available, an utmost priority task with the greatest algorithmic parameter is submitted to execution queue.

It should be noted that Grid uses peer-to-peer communication model [30, 33] to save time in transferring data. Peer-to-peer communication provides a means to transfer output of an executed task directly (without mediator) from the Grid resource on which the task is

Figure 5
The Workflow Management System


DAG MANAGER
TASK SCHEDULER
WORKFLOW ARRIVAL
MONITOR FOR SUBMITED WORKFLOW
MONITOR FOR TASKS TERMINATION
READY TASKS
RECEIVING SORTED, READY TASKS AND MONITORING TASKS TERMINATION
INCLUDE WORKFLOW TO DAG
IS THERE A WAITING TASK FOR AVAILABLE RESOURCE?
YES
NO
SEND TASK TO EXECUTION
SEND TASKS TO EXECUTION
GRID INFRASTRUCTURE
REMOVE TERMINATED TASKS FROM DAG
ANALYZE DAG
ARE THERE READY TASKS?
NO
YES
CALCULATE ALGORITHMIC PARAMETER
SORT READY TASK ACCORDING TO PRIORITY, COMPUTATIONAL RESOURCE AND ALGORITHMIC PARAMETER
ESTIMATE WORKLOAD OF COMPUTING SERVICES
ARE THERE AVAILABLE RESOURCES?
NO
YES
SEND TASKS TO EXECUTION QUEUE

completed to the Grid resource where the successor of the task has to be executed. However, *when a successor task is a computing task which has to be executed on a CS, and since this type of DMS service is hosted on multiply computational resources, the WMS has to resolve to which CS should output data be transferred.* Hence, if any of the ready tasks which are sent to execution have a computing task as a successor, the engineered WMS determines the most appropriate CS for executing the successor computing task. Therefore, the ready task and the CS address are sent to Grid. Consequently, the output of the ready task is directed to the submitted address. To ensure this behavior and support beneficial scheduling, the DAG Manager provides the following *estimation of CSs workloads*:

1 After the classification, ready tasks with the highest priority and the greatest rank are extracted and further analyzed (it is expected that these tasks will be executed first). Computing ready tasks are excluded from this step.
2 If any of the extracted tasks have computing tasks as successors, appropriate CSs have to be determined for each computing successor, so that addresses of suitable CSs may be submitted together with non-computing ready tasks. This approach ensures that outputs of non-computing task (which have computing tasks as successors) are transferred to previously determined CSs.
3 The workload of CSs is examined (centralized WMS is aware of computing tasks already assigned to CSs). The successor computing tasks with the highest priority and the largest rank are mapped to the CSs with the lowest workload.
4 It is obvious that ready computing tasks are excluded from analysis in Step 1 since they are already mapped to CSs (during Step 3 in one of the previous scheduling). During the previous scheduling, they were successors of one or more ready tasks.

The estimation of CSs workload is executed only for centralized scheduling, hence it is displayed as a dashed square in Figure 5. When a CS becomes available, the DAG Manager chooses *the most eligible* computing task (a computing task with the highest priority and the greatest rank). It is chosen from the set of ready computing tasks previously mapped to the CS during the estimation of CSs workload phase. The outputs of the predecessor of this task are directly transferred to the available CS.

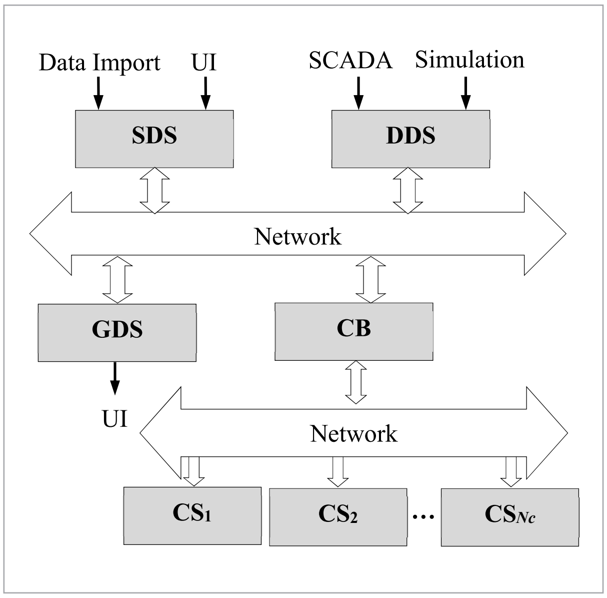
4.1.3. The Task Scheduler

The Task Scheduler is responsible for submitting tasks to the Grid resources and monitoring task completion. The scheduler maintains the task queue which is constantly read. The tasks are independent and already mapped to appropriate computational resources.

4.2. The Mixed Scheduling Concept

By using the mixed scheduling concept, scheduling of computing tasks is improved at the expense of losing some transfer-time gaining achieved through peer-to-peer communication. An additional distributed scheduling [1] is included and it exclusively manages ready computing task. The aim of distributed scheduling is to achieve parallelism through the workload distribution and obtain better execution time. A more advanced computing engine includes multiple CSs and establishes the Computing Broker (CB) (see Figure 6).

Figure 6
Modification of the Grid for mixed scheduling



CB represents a mediator between multiple CSs and the other parts of the system. The drawback of this approach is that direct peer-to-peer communication between the CSs and other resources (DMS services) does not exist.

The CSs implement a distributed scheduling algorithm, hence, the estimation of CSs workload as a part of centralized scheduler becomes redundant. The CSs become more complex considering that in addition to executing computing tasks, they are performing workload balancing. Now WMS consists of two schedulers, the main centralized scheduler and the distributed scheduler for computing tasks.

The centralized scheduler continues to analyze DAG, calculates algorithmic parameters and classifies ready tasks. Non-computing tasks are scheduled in the orderly manner – the most eligible task is sent to the available resource. However, scheduling computing tasks is now a responsibility of the advanced computing engine, and as soon as a computing task becomes ready, it is transferred to CB. CB receives ready computing tasks from the Task Scheduler and input data from other Grid resources. When a CS adopts a task, it takes data contained by CB and executes expected calculation. After completing the task, the CS transfers results to CB, which forwards them to the appropriate Grid resource and notifies the centralized scheduler (more accurate, the Task Scheduler) of the task completion.

4.2.1. The Distributed Algorithm

It is of the most significance that CSs share information about their current workload, thus the cooperative scheduling is provided [14, 31]. By ensuring cooperative operation, a CS makes decisions regarding the scheduling in line with other CSs. Hence, each CS is responsible to carry out its own part of scheduling work which will contribute to achieving the common global goal – workload distribution according to the capacity of computational resources.

NC is defined as the set of all CSs within a Grid. The cardinality of *NC* is denoted by N_c. If n_i indicates a CS, nw_{n_i} and np_{n_i} represent the current workload and computational capacity of resource n_i, respectively. Accordingly, the desired workload distribution over the CSs is determined:

$$\overline{nw}_{n_i} = \frac{\sum_{n_j \in NC} nw_{n_j}}{\sum_{n_j \in NC} np_{n_j}} \cdot np_{n_i}. \quad (7)$$

The aim of distributed scheduling is to make workload distribution converge towards the desired distribution:

$$nw_{n_i} \to \overline{nw}_{n_i}, \qquad \forall n_i \in CN. \quad (8)$$

The distributed algorithm is an iterative algorithm and it is based on the physical phenomenon of diffusion. As matter is moved from a region of high concentration to a region of low concentration, computing tasks similarly migrate from a more loaded CS to a less loaded CS.

The algorithm makes decisions about workload distribution at the iteration h by using the workload information of the iteration $h - 1$ [12]. $nw^h_{n_i}$ represents the current workload of computing service n_i at iteration h (sum of workloads (w) of all computing tasks currently assigned to n_i). According to the diffusion algorithm, the CS marked with n_i exchanges a share of workload $\alpha \cdot \left| nw^h_{n_i} - nw^h_{n_j} \right|$ with another CS labeled as n_j at each iterative step, where parameter α determines how much of workload difference is exchanged [12, 40].

Workload modification of n_i after the workload exchange with all CSs is described by the following equation:

$$nw^h_{n_i} = nw^{h-1}_{n_i} + \sum_{n_j \in NC} \alpha \cdot [nw^{h-1}_{n_j} - nw^{h-1}_{n_i}]. \quad (9)$$

By rearranging Equation (9), the following form is obtained:

$$nw^h_{n_i} = [1 - \alpha \cdot N_c] \cdot nw^{h-1}_{n_i} + \alpha \cdot \sum_{n_j \in NC} nw^{h-1}_{n_j}. \quad (10)$$

Equation (10) is linear, therefore the iterative distribution algorithm may be presented in matrix form:

$$\mathbf{NW}^h = \mathbf{DM} \cdot \mathbf{NW}^{h-1}, \quad (11)$$

where $\mathbf{NW}^h$ is the workload distribution of CSs at iteration *h:*

$$\mathbf{NW}^h = [nw^h_{n_1}, nw^h_{n_2}, \ldots, nw^h_{n_{N_c}}], \quad (12)$$

and **DM is a diffusion matrix with the following entries:**

$$dm_{ij} = \begin{cases} \alpha, & i \neq j \\ 1 - \alpha \cdot N_c, & i = j \end{cases} . \quad (13)$$

Completion of the algorithm is related to the possibility to lead any initial workload distribution to the workload balanced state.

4.2.2. The Implementation of the Distributed Algorithm by CSs

CSs exchange messages about current workload so that cooperative scheduling is achieved. It is unnecessary to exchange information when the workload balance exists – an appropriate amount of workload is assigned to each CS. When the balance is disturbed, it is essential to run the algorithm to re-establish the balance. Therefore, the algorithm starts on the occurrence of the following events:

- When a CS receives a new computing task (just arrived from CB), it considers that the existing workload balance is disturbed. The CS starts information exchange to find out if any other CS is less loaded.
- When a computing task (an amount of workload) is transferred from a CS to another, less loaded CS, the workload of the second CS has just increased. The CS which has received the workload, acknowledges the imbalance and searches for other CSs which are able to collect some of its workload.
- When a new CS is added to the computational engine, the remaining CSs examine how much of their workload may be taken over by the new CS.

A CS considers that balance is established when there is no need for further workload exchange between it and other CSs.

While exchanging workload information, the CSs simultaneously execute tasks. A CS chooses the most eligible computing task (according to the priority and algorithmic parameter) assigned to it to be completed first. Tasks which are currently executed on CSs cannot migrate during the run of the distributed algorithm.

4.3. The Hierarchical Scheduling Concept

Hierarchical scheduling is applied on the Grid presented in Figure 6. This scheduling concept is much similar to the mixed scheduling, except that workload distribution is not done by cooperative work of CSs, but by a single CB. In this concept, CB receives independent computing tasks, keeps records about workload and maps tasks to the CSs. As a consequence, the role of CB is dual:

- It hides multiplicity of CSs from the other computational resources, so the peer-to-peer communication with the CSs is disabled.
- It operates as an autonomous, low-level computing tasks scheduler. It uses the same rules explained in the section about the mixed scheduling concept.

The most eligible computing tasks have the advantage during assignment of tasks to the CSs. By supporting this behavior, the low-level scheduler of computing tasks takes into consideration influence of each independent computing task on the makespan of the whole workflow application.

5. The Experimental Study and Discussion

This section explains the conducted experiments whose results are further used to elaborate on comparison of presented scheduling concepts.

5.1. Experiments

A testing environment with four CSs [25] is developed to simulate operation of the DMS during a real-time DN control. Computational resources used in test environment have the following characteristics:

- SDS: Intel(R) Core(TM) i3-3222 CPU 3.30 GHz, 16 GB RAM, SQL Server 2012,
- DDS: Intel(R) Core(TM) i3-4130 CPU 3.40 GHz, 24 GB RAM,
- GDS: Intel(R) Core(TM) i3-4150 CPU 3.50 GHz, 32 GB RAM,
- Four CSs: Intel(R) Core(TM) i3-4160 CPU 3.60 GHz, 24GB RAM.

Software components are implemented in .NET 4.5 Framework.

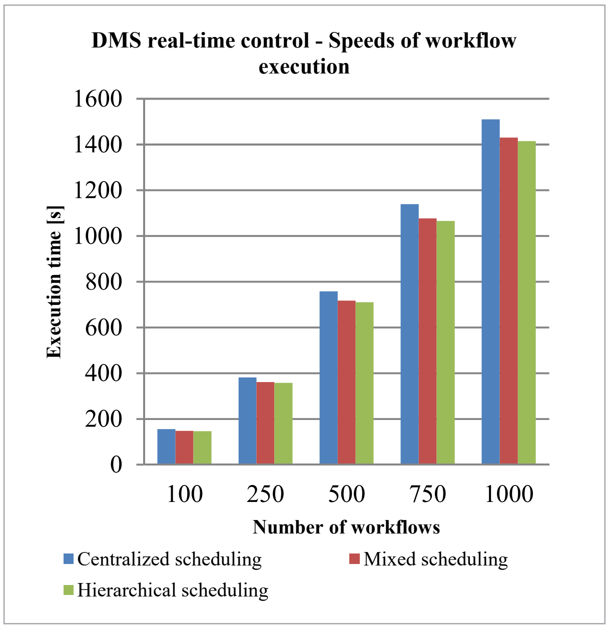
All experiments examine the effect of makespan changes which are used for comparison of the presented scheduling concepts. The workflow application is composed of the all considered DMS workflow types (Figure 2). DMS Function Execution is the most common and it makes more than 70%, while the Model Update workflow type represents about 20% of all workflows. The Graphical View Refresh type is the most infrequent (up to 10% of all workflows). Between a hundred and a thousand workflows arriving dynamically are used in experiments. To achieve the highest possible workload sharing between CSs, parameter α is set to $1/2$ ($\alpha = 1/2$), in accordance with [12, 40].

The obtained results, total speeds of workflow execution, are presented in Table 1 and Figure 7.

Table 1
The speeds of workflow execution

Number of workflows	Centralized scheduling [s]	Mixed scheduling [s]	Hierarchical scheduling [s]
100	155.65	147.66	146.17
250	381.38	361.30	357.52
500	757.65	717.59	710.04
750	1139.63	1076.75	1065.54
1000	1510.15	1430.09	1415.05

Figure 7
DMS real-time control – Workflow execution speeds

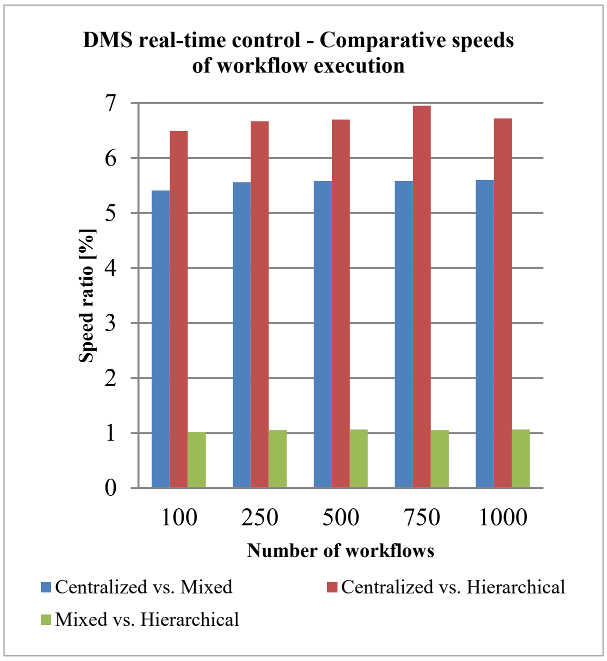


The comparative speeds of different workflow scheduling concepts are shown in Table 2 and Figure 8.

Table 2
The comparative speeds of workflow execution

Number of workflows	Centralized v.s. Mixed [%]	Centralized v.s. Hierarchical [%]	Mixed v.s. Hierarchical [%]
100	5.41	6.49	1.02
250	5.56	6.67	1.05
500	5.58	6.70	1.06
750	5.58	6.95	1.05
1000	5.60	6.72	1.06

Figure 8
DMS real-time control – Comparative speeds of workflow execution



The experimental study leads to the following conclusions:

_ The mixed and the hierarchical scheduling exceed centralized concept.

_ The scheduling brings more benefits if there are more workflows to coordinate.

_ The hierarchical scheduling concept slightly outperforms the mixed solution.

_ The time savings increase by increasing the number of workflows.

5.2. Discussion

The presented scheduling concepts are compared by different parameters: performance, scalability and complexity of implementation.

5.2.1. Performance

The utilization of two schedulers, rather than one centralized, reduces the total execution time and boosts the performance of the whole system. The centralized scheduling pre-determines the mapping of computing tasks to CSs and that feature can be the main reason for the weaker performances. A computing task is always executed to a previously assigned resource regardless the current state of the unpredictable resource (e.g. the resource may have been occupied with some internal tasks).

The hierarchical scheduling concept exceeds the mixed concept to a certain extent. This can be explained by the fact that CSs are not required to perform workload distribution during the hierarchical scheduling. They entirely devote time to execute computing tasks.

5.2.2. Scalability

Since DMS Function is by far the most frequent DMS workflow type, scalability is considered in the context of computing tasks. For a large-scale DMS, it can be essential to include additional CSs in the advanced computing engine, so that execution of computing tasks would not be the bottleneck of the system.

The centralized scheduling concept is the least scalable. An addition of a new CS requires changes in all system components: the DAG Manager and the Task Scheduler have to provide scheduling for the new CS, and all other computational Grid resources must obtain the address of the CS.

The implementation of the mixed scheduling generates the need that CB and all existing CSs have to incorporate the new CS into the distributed algorithm.

The hierarchical scheduling concept is the most scalable. The new CS is introduced only by connecting to CB.

5.2.3. The Complexity of Implementation

The centralized scheduling holds the complete set of information. It is typical for this scheduling concept that decision making is carried out in one fundamental place. Thus, easy implementation is an advantage of this concept. The replacement of scheduling algorithm in the presented centralized scheduling is simply manageable by changing the definition of algorithmic parameter.

The hierarchical scheduling is composed of two schedulers, a centralized and a low-level. A benefit is that two schedulers may use different algorithms. Both implementations are straightforward, but supporting this concept requires more effort than supporting purely centralized scheduling.

The mixed scheduling concept is also composed of two schedulers, a centralized scheduler and a low-level distributed scheduler for computing tasks. This concept requires the most elaborate implementation. In addition to the mixed scheduler, each CS must provide dual features – task execution and workload exchanging.

6. Conclusion

The paper has discussed three scheduling concepts that take into account dynamic nature of DMS and specific characteristics of DMS workflows: centralized, mixed and hierarchical. DMS grid architecture and architecture-dependent implementation was explained for each of them. The experimental study was conducted to examine their effectiveness and benefit which they provide during DMS operation in control mode.

The workflow scheduling concepts are mutually compared based on the scheduling efficiency (makespan), scalability and complexity of implementation. Taking into consideration observations regarding these parameters the concept of hierarchical scheduling emerges as the finest choice since it ensures the greatest performance, it is the most scalable and completely suitable in terms of the complexity of implementation.

References

1. Arora, M., Das, S. K., Biswas, R. A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments. International Conference on Parallel Processing Workshops, Vancouver, Canada, 2002, 499-505. https://doi.org/10.1109/ICPPW.2002.1039771

2. Baker, M., Buyya, R., Laforenza. D. Grids and Grid Technologies for Wide-Area Distributed Computing. Software: Practice and Experience, 2002, 32(15), 1437-1466. https://doi.org/10.1002/spe.488

3. Bose, A. Smart Transmission Grid Applications and Their Supporting Infrastructure. IEEE Transactions on Smart Grid, 2010, 1(1), 11-19. https://doi.org/10.1109/TSG.2010.2044899

4. Braun, T. D., Siegal, H. J., Beck, N., Boloni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D., Freund, R. F. A Comparison Study of Static Mapping Heuristics for a Class of Meta-Tasks on Heterogeneous Computing Systems. Heterogeneous Computing Workshop, San Juan, Puerto Rico, 1999, 15-29. https://doi.org/10.1109/HCW.1999.765093

5. Cao, J., Jarvis, S.A., Saini, S., Nudd, G.R. Grid Flow: Workflow Management for Grid Computing. Proceedings of International Symposium on Cluster Computing and the Grid, (CCGrid), Tokyo, Japan, 2003, 198-205. https://doi.org/10.1109/CCGRID.2003.1199369

6. Capko, D., Erdeljan A., Popovic M., Svenda, G. An Optimal Initial Partitioning of Large Data Model in Utility Management Systems. Advances in Electrical and Computer Engineering, 2011, 11(4), 41-46. https://doi.org/10.4316/AECE.2011.04007

7. Capko, D., Erdeljan A., Vukmirovic S., Lendak, I. A Hybrid Genetic Algorithm for Partitioning of Data Model in Distribution Management Systems. Information Technology and Control, 2011, 40(4), 316-322. https://doi.org/10.5755/j01.itc.40.4.981

8. Casanova, H., Legrand, A., Zagorodnov, D., Berman, F. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. The 9th Heterogeneous Computing Workshop, (HCW'00), Cancun, Mexico, 2000, 349-363. https://doi.org/10.1109/HCW.2000.843757

9. Cassel, W. R. Distribution Management Systems: Functions and Payback. IEEE Transactions on Power Systems, 1993, 8(3), 796-801. https://doi.org/10.1109/59.260926

10. Chapin, S. J., Katramatos, D., Karpovich, J., Grimshaw, A. S. The Legion Resource Management System. Proceedings of Job Scheduling Strategies for Parallel Processing, (JSSPP'99), San Juan, Puerto Rico, 1999, 162-178. https://doi.org/10.1007/3-540-47954-6_9

11. Chen, H., Maheswaran, M. Distributed Dynamic Scheduling of Composite Tasks on Grid Computing Systems. Proceedings of Parallel and Distributed Processing Symposium, (IPDPS 2002), Fort Lauderdale, USA, 2002, 88-97. https://doi.org/10.1109/IPDPS.2002.1015664

12. Cybenko, G. Dynamic Load Balancing for Distributed Memory Multiprocessors. Journal of Parallel and Distributed Computing, 1989, 7(2), 279-301. https://doi.org/10.1016/0743-7315(89)90021-X

13. Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Blackburn, K., Lazzarini, A., Arbree, A., Cavanaugh, R., Koranda, S. Mapping Abstract Complex Workflows onto Grid Environments, Journal of Grid Computing, 2003, 1(1), 25-39. https://doi.org/10.1023/A:1024000426962

14. Dong, F., Akl, S. G. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Technical Report No. 2006-504, Kingston, Canada, 2006.

15. Foster, I., Kesselman, C., Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organization. The International Journal of High Performance Computing Applications, 2001, 15(3), 200-222. https://doi.org/10.1177/109434200101500302

16. Hamscher, V., Schwiegelshohn, U., Streit, A., Yahyapour, R. Evaluation of Job-Scheduling Strategies for Grid Computing. Proceedings of International Workshop on Grid Computing, (Grid 2000), Heidelberg, Germany, 2000, 191-202. https://doi.org/10.1007/3-540-44444-0_18

17. Harel, D., Pnueli, A. On the Development of Reactive Systems, Logics and Models of Concurrent Systems. Springer-Verlag New York, New York, USA, 1985, 477-498. https://doi.org/10.1007/978-3-642-82453-1_17

18. Hotovy, S. Workload Evolution on the Cornell Theory Center IBM SP2. Workshop on Job Scheduling Strategies for Parallel Processing, Honolulu, Hawaii, USA, 1996, 27-40. https://doi.org/10.1007/BFb0022285

19. Iverson, M., Ozguner, F. Dynamic, Competitive Scheduling of Multiple DAGs in a Distributed Heterogeneous Environment. Heterogeneous Computing Workshop, Orlando, Florida, USA, 1998, 70-78. https://doi.org/10.1109/HCW.1998.666546

20. Kaceniauskas, A. Solution and Analysis of CFD Applications by Using Grid Infrastructure. Information Technology and Control, 2010, 39(4), 284-290.

21. Kumar, P., Anandarangan, V., Reshma, A. An Approach to Workflow Scheduling using Priority in Cloud Computing Environment, International Journal of Computer Applications, 2015, 109(11), 32-38. https://doi.org/10.5120/19236-1008

22. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J., Zhao, Y. Scientific Workflow Management and the Kepler System, Concurrency and Computation: Practice and Experience, 2006, 18(10), 1039–1065. https://doi.org/10.1002/cpe.994

23. Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., Freund, R. F. Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. Journal of Parallel and Distributed Computing, 1999, 59(2), 107-131. https://doi.org/10.1006/jpdc.1999.1581

24. Mateescu, G. Quality of Service on the Grid via Metascheduling with Resource Co-Scheduling and Co-Reservation. International Journal of High Performance Computing Applications, 2003, 17(3), 209-218. https://doi.org/10.1177/1094342003173006

25. Nedić, N., Švenda, G. Workflow Management System for DMS. Information Technology and Control, 2013, 42(4), 373-385. https://doi.org/10.5755/j01.itc.42.4.4546

26. Nedic, N., Vukmirovic, S., Erdeljan, A., Imre, L., Capko, D. A Genetic Algorithm Approach for Utility Management System Workflow Scheduling. Information Technology and Control, 2010, 39(4), 310-316.

27. Park, D. Concurrency and Automata on Infinite Sequences. Proceedings of GI-Conference on Theoretical Computer Science, London, UK, 1981, 167-183. https://doi.org/10.1007/BFb0017309

28. Sakellariou, R., Zhao, H. A Low-Cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems. Scientific Programming, 2004, 12(4), 253-262. https://doi.org/10.1155/2004/930924

29. Santacana, E., Rackliffe, G., Tang, L., Feng, X. Getting Smart. IEEE Power and Energy Magazine, 2010, 8(2), 41-48. https://doi.org/10.1109/MPE.2009.935557

30. Schollmeier, R. A Definition of Peer-to-peer Networking for the Classification of Peer-to-Peer Architectures and Applications. Proceedings of International Conference on Peer-to-Peer Computing, (P2P'01), Washington, DC, USA, 2001, 101-102. https://doi.org/10.1109/P2P.2001.990434

31. Shan, H., Oliker, L., Biswas, R., Smith, W. Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration. Proceedings of Advanced Computing and Communication, (ADCOM 2004), Ahmedabad Gujarat, India, 2004, 1-8. https://doi.org/10.2172/860301

32. Siegel, H. J., Ali, S. Techniques for Mapping Tasks to Machines in Heterogeneous Computing Systems. Journal of Systems Architecture, 2000, 46, 627-639. https://doi.org/10.1016/S1383-7621(99)00033-8

33. Singh, M. P. Peering at Peer-to-Peer Computing. IEEE Internet Computing, 2001, 5(6), 4-5. https://doi.org/10.1109/MIC.2001.968826

34. [34] Son, J. H., Kim, M. H. Analyzing the Critical Path for the Well-Formed Workflow Schema. Database Systems for Advanced Applications, Hong Kong, China, 2001, 146-147. https://doi.org/10.1109/DASFAA.2001.6044749

35. Švenda, G., Strezoski, V., Kanjuh, S. Real-Life Distribution State Estimation Integrated in the Distribution Management System. International Transactions on Electrical Energy Systems, 2016, 27(5), 1-16. https://doi.org/10.1002/etep.2296

36. Topcuoglu, H., Hariri, S., Wu, M. Y. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(3), 260-274. https://doi.org/10.1109/71.993206

37. Van der Aalst, W. M. P., Ter Hofstede, A. H. M., Kiepuszewski, B., Barros, A. P. Workflow Patterns. Technical Report, Eindhoven University of Technology, 2000.

38. Varga, E., Lendak, I., Gavric, M., Erdeljan, A. Applicability of RESTful Web Services in Control Center Software Integrations. International Conference on Innovations in Information Technology – IIT, Abu Dhabi, United Arab Emirates, 2011, 282-286. https://doi.org/10.1109/INNOVATIONS.2011.5893833

39. Vukmirovic, S., Erdeljan, A., Lendak, I., Capko, D., Nedic, N. Optimization of Workflow Scheduling in Utility Management System with Hierarchical Neural Network. International Journal of Computational Intelligence Systems, 2011, 4(4), 672-679. https://doi.org/10.2991/ijcis.2011.4.4.22

40. Xu, C., Lau, F. Optimal Parameters for Load Balancing with the Diffusion Method in Mesh Networks. Parallel Processing Letters, 1994, 4(2), 139-147. https://doi.org/10.1142/S0129626494000156

41. Zhang, Y., Koelbel, C., Cooper, K. Hybrid Re-scheduling Mechanisms for Workflow Applications on Multi-cluster Grid. Cluster Computing and the Grid, (CCGRID'09), Shanghai, China, 2009, 116-123. https://doi.org/10.1109/CCGRID.2009.60