CMST 17(1) 53-62 (2011)
DOI:10.12921/cmst.2011.17.01.53-62

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY **17(1-2)**, 53-62 (2011)

Computational Experiments for Scheduling Workflow Applications in Grid Environment

Marek Mika[1], Wojciech Piątek[2], Grzegorz Waligóra[1], Jan Węglarz[1,2]

[1]*Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60-965 Poznań, Poland*
e-mail: {marek.mika/grzegorz.waligora/jan.weglarz}@cs.put.poznan.pl
[2]*Poznan Supercomputing and Networking Center, Noskowskiego 12/4, 61-704 Poznań, Poland*
e-mail: piatek@man.poznan.pl

(Received: 17 November 2011; accepted: 5 December 2011; on-line: 23 December 2011)

Abstract: The problem of scheduling workflow applications in a grid environment is considered. The problem is divided into two stages: (i) resource allocation, which consists in allocating distributed grid resources to tasks of a workflow in such a way that the resource demands of each task are satisfied, and (ii) scheduling performed by local grid schedulers. Grid resources are divided into computational and network resources. Computational and transmission workflow tasks are distinguished. A computational experiment is presented in order to show the importance of resource allocation, as well as examine the influence of the local scheduling policy. Certain conclusions and directions for future research are given.

Key words: grid, workflow, resource allocation, scheduling

I. INTRODUCTION

In this paper, the problem of scheduling workflow applications in a grid environment is considered. Workflow applications can be viewed as complex sets of various precedence-related transformations (tasks) performed on certain data. They are mostly scientific, data-intensive applications which, due to large amounts of computations and data involved, require high computing power to be executed efficiently. This power can be delivered by a grid, an infrastructure consisting of many distributed computational resources connected by a fast network.

The problem of allocating distributed grid resources, located in many sites, to workflow application tasks is very complex, especially when the network capacity varies between the sites. In addition, the information about the tasks is often incomplete. Obtaining a performance model of a task is not trivial. In particular, the processing times of all tasks on different computer systems (grid resources) are difficult to evaluate. Also, other parameters (e.g. bandwidth, resource availability) may change quite rapidly in grid environments. Thus, generally, many problem parameters are dynamic and/or uncertain.

As was stressed, e.g. in [5], the majority of centralized grid environments are based on the so-called two-level hierarchy scheduling. This is a consequence of the layered architecture of the grid. A grid consists of many nodes, each of which is usually managed by some local scheduling system, such as Condor [7], Load Sharing Facility (LSF) [8], Portable Batch System (PBS) [9], Sun Grid Engine (SGE) [10]. Thus, in the first step, the grid broker assigns submitted jobs to remote resources, and local schedulers subsequently generate their schedules for the resources they manage. This concept is very natural, since a grid broker neither possesses complete knowledge of the local resource load, nor has overall control of the resource. On the other hand, the local scheduler is unaware of any other grid jobs and other resources available to these jobs. Examples of two-level hierarchy grids with a central grid broker are, among others, the European EGEE Grid [11], and Clusterix in Poland [12]. Let us emphasize the difference between resource allocation and scheduling. Resource allocation consists in assigning tasks to resources, whereas scheduling goes one step further and consists in allocating resources to tasks over time, i.e. defining the starting time of each task for the resource to which it has been assigned.

Due to the two-level hierarchy scheduling, the first stage, i.e. the grid resource allocation, is of crucial importance, in particular when it concerns workflow applications requiring substantial computational effort and can run for hours, days, or even weeks.

In [6], we proposed a model of the problem where the network is a resource for which tasks must apply. Consequently, we distinguished transmission tasks as a separate type of tasks which can compete for the same network connection. More precisely, bandwidth is a network resource which can be divided among many transmission tasks. Thus, on a grid, there are network resources along with computational resources. A mathematical model of the problem was presented along with all the parameters of the problem. However, in [6] we focus only on resource allocation, and present algorithms for finding feasible resource allocations. In this paper, we go one step further and perform certain computational experiments concerning scheduling workflows on a grid. The main goal of this paper is to show the importance of resource allocation on the quality of the obtained schedules, as well as examine the influence of the local scheduling policy.

The paper is organized as follows. In Section II, we define the problem and recall its parameters introduced in [6]. Section III discusses the phases of resource allocation and local scheduling. In Section IV, we describe the computational experiments and analyze the results. The final section is devoted to conclusions and describes potential directions for future research.

II. PROBLEM FORMULATION

II.1. Problem description

In many scientific areas, such as high-energy physics, bioinformatics, astronomy and others, we encounter applications involving numerous simpler components that process large data sets, execute scientific simulations, and share both data and computing resources. Such data-intensive applications consist of multiple components (tasks), which may communicate and interact with each other over the course of the application [2]. The tasks are very often precedence-related, and the precedence constraints usually follow from the data flow between them, i.e. data files generated by one task are often needed to start another task (an output of one task becomes an input for the next task). Although this is the most common situation, the precedence constraints may be dictated by other considerations, for instance, they may be arbitrarily defined by the user. Such complex applications consisting of various precedence-related transformations (tasks) performed on certain data between which data files have to be transmitted very often are called workflow applications. In general, two types of workflows can be distinguished: data-intensive, where file transfer times dominate task computing times, and compute-intensive, for which the situation is opposite. For efficient execution of both types of workflows, high computing power is required, due to a large amount of computations and data involved. This computational power can be provided by a grid.

In [3], a grid was defined as an infrastructure for coordinated resource sharing and problem solving in dynamic, multi-instrumental virtual organizations. More recently, the Network of Excellence CoreGRID [13] defined a grid as "a fully distributed, dynamically reconfigurable, scalable and autonomous infrastructure to provide location independent, pervasive, reliable, secure and efficient access to a coordinated set of services encapsulating and virtualizing resources (computing power, storage, instruments, data, etc.) in order to generate knowledge".

Since workflow applications are usually very time-consuming (even if single tasks are short) and input/output data files for tasks can be large, the problem of scheduling such applications on a grid has nowadays become a significant challenge of great practical importance. As it was mentioned in the Introduction, resource allocation must precede scheduling. Allocation of grid resources to a workflow whose component tasks are known but have not been scheduled yet is an important topic in grid computing due to its impact on the efficiency of workflows, which can generate great amounts of data and occupy resources for longer periods of time.

In general, the problem consists in allocating distributed grid resources to heterogeneous tasks. Users submit their jobs (in this case, workflow applications) to a grid. Workflows consist of multiple tasks. A task can be anything that requires a resource, e.g. schedulable computation, bandwidth request, data access or access to a remote resource, such as remote instruments, databases, humans in the loop, etc. A resource is anything that can be allocated to a task, i.e. a processor, disc space, bandwidth, machine, device, person, etc. In this research, we divide grid resources into two types: computational resources and network resources. Computational resources are all resources required for computational tasks to be computed, i.e. not only processors, but also memory, disc space, various devices, etc. Network resources are resources required for transmission tasks to be executed, i.e. necessary for the data files to be transmitted. Obviously, the basic network resource is bandwidth.

There are at least several approaches to grid resource allocation. They differ from one another depending on the grid architecture, objectives of a particular grid, and grid management policies. Depending on the architecture, two types of grids can be distinguished: peer-to-peer grids, where all services are equal and communicate using a peer-to-peer model; and centralized grids, where a grid resource management system plays a central role and is surrounded by many other grid services structured in a layered architecture. In such grids, there is usually one common, central grid broker that serves all users and their jobs. Such a situation is considered in this paper.

II.2. Problem parameters

In this section we briefly recall the main assumptions underlying the model presented in [6], as well as present all the parameters of the formulated model.

A grid is a set of nodes connected by fast network links. There are two types of nodes in the network: resource nodes (containing computational resources) and non-resource nodes (considered only with respect to the network topology). The bandwidth between each two connected nodes is given, and it is identical in both directions. Between two given nodes, there can be more than one network link, and these links may have different parameters. However, these are alternative links and they cannot be merged in order to increase the bandwidth. The bandwidth within a given node is unlimited. The processors in resource nodes are divided into different types, depending on their power. The power (processing speed) of each processor is a multiple of some standard unit. Here, a processor with a speed factor of 1 is termed a standard processor.

A workflow consists of many tasks. There are two types of tasks: computational tasks and transmission tasks. The structure of a workflow is represented by a directed acyclic graph (DAG), where each vertex corresponds to a computational task, and each arc represents a precedence relation between two computational tasks. Each arc corresponds to a transmission task, i.e. represents a data transmission between two successive computational tasks. Computational tasks are non-preemptable, i.e. once started, they must be completed without interruptions or changes in resource allocation. Each computational task may be executed in exactly one node, and is characterized by three values: size (the execution time on a standard processor or processors), number of processors required for its execution, and the minimum required speed factor of the processors. The actual processing time of a task is calculated by dividing its size by the speed factor of the processor (processors) on which this task is scheduled. Transmission tasks are also non-preemptable, and they are characterized by two values: the size of the data file (files) to be transmitted, and the required bandwidth between the two nodes between which the transmission is to take place. Transmission time (i.e. the execution time of a transmission task) can take one of two values: the data file size divided by the bandwidth, when successive computational tasks are executed in different resource nodes; or zero, when they are executed in the same resource node.

Below we summarize all the parameters of the model:

A) Grid

$\Gamma(N,\Psi)$ – undirected multigraph representing the network topology of a grid;

N – set of all (resource and non-resource) nodes in the network, $N = X \cup \Pi$

X – set of resource nodes;

Π – set of non-resource nodes;

Ψ – set of edges (links) between nodes, i.e. set of pairs $(\mu,\nu)_\psi : \mu,\nu \in N,\ \psi = 1,2...$ (ψ denotes alternative links between a given pair of nodes);

X_χ – resource node χ, $\chi = 1,2,...,|X|$;

Π_η – non-resource node η, $\eta = 1,2,...,|\Pi|$;

ϖ_κ – speed factor for processors of type κ; $\kappa = 1, 2 ...$;

$P_{\kappa\chi}$ – number of processors of type κ; $\kappa = 1, 2 ...$ (i.e. processors with the same speed factor ϖ_κ in resource node X_χ;

$\Psi_\psi^{\mu\nu}$ – bandwidth of the link $(\mu,\nu)_\psi \in \Psi$, $\psi = 1,2,...$;

B) Workflow

$W(V, E)$ – directed acyclic graph representing the structure of the workflow;

V – set of vertices of graph W representing computational tasks;

E – set of arcs (v_i, v_j) of graph W representing precedence constraints between computational tasks $v_i, v_j \in V$, i.e. transmission tasks;

Computational tasks

v_i – computational task $i, i = 1, 2, ..., |V|$;

p_i – size of computational task v_i (expressed in assumed computational units, e.g. MIPS or similar), $i = 1, 2, ..., |V|$;

r_i – number of processors required for the execution of computational task $v_i, i = 1, 2, ..., |V|$;

ω_i – minimum speed factor of the processor (processors) required for the execution of computational task $i = 1,2,...,|V|$;

$f_i(p_i,\varpi_k)$ – function defining the actual execution time of computational task v_i $(i = 1,2,...,|V|)$ on processor (processors) with speed factor ϖ_κ $(\varpi_\kappa \geq \omega_i)$; we assume that $f_i = f = p_i / \varpi_k$, $(i = 1, 2, ..., |V|;\ \kappa = 1, 2, ...)$;

Transmission tasks

(v_i, v_j) – task of transmitting output data of computational task v_i, which is at the same time the input data for computational task $v_j (v_i, v_j) \in E$;
F^{ij} – size of data file (files) transmitted as a result of the execution of transmission task (v_i, v_j);
B^{ij} – minimum required bandwidth of the connection between resource nodes in which computational tasks v_i and v_j are executed;
$g(F^{ij}, B^{ij})$ – execution time of transmission task (v_i, v_j) i.e. the time of data transmission from the resource node in which computational task v_i is executed to the resource node in which computational task v_j is performed. We assume that $g(F^{ij}, B^{ij}) = F^{ij} / B^{ij}$, if tasks v_i and v_j are executed in different nodes; or is equal to 0, if they are executed in the same node.

III. RESOURCE ALLOCATION AND SCHEDULING

As it was mentioned before, scheduling in grid environments consists of two stages:

(i) resource allocation performed by a central grid broker on the set of all resources available in the grid
(ii) scheduling performed by local schedulers on their local resources.

The problem of resource allocation in grids was extensively studied in [6], where a mathematical model of the problem was formulated and algorithms for finding a feasible resource allocation were proposed. The most important issues raised in [6] are briefly recalled in Sections III.1. and III.2, which concern the second (scheduling) stage, as well as describe how this stage was realized in the Grid Scheduling Simulator (GSSIM).

III.1. Resource allocation

By feasible resource allocation we understand such an allocation of resource nodes to computational tasks of a workflow that:

– each computational task is assigned resource nodes containing a computational resource capable of executing this task,
– for each transmission task there exists a path in the grid over which the required transmission can be performed (i.e. a path of the required bandwidth).

In order to analyze and correctly define feasible resource allocations, a concept of a tri-task was introduced in [6]. A tri-task $\langle v_i, v_j \rangle$ is a triple $\{v_i (v_i, v_j), v_j\}$, i.e. two computational tasks and a transmission task between them. Therefore, feasible resource allocation RA^{ij} for a tri-task $\langle v_i, v_j \rangle$ is a pair of nodes (X_χ, X_θ), so that the task v_i can be performed in node X_χ, task v_j can be performed in node X_θ, and the transmission task (v_i, v_j) can be performed between nodes X_χ and X_θ. In [6], an algorithm was presented for finding the set A^{ij} of all feasible resource allocations for a tri-task $\langle v_i, v_j \rangle$ if at least one such allocation exists. This algorithm was termed the RA_{TT} algorithm. Subsequently, another algorithm was proposed, which uses the RA_{TT} algorithm as a sub-algorithm for finding feasible resource allocation RA_W for the entire workflow W. A feasible resource allocation RA_W is defined by a function w assigning exactly one node to each computational task in such a way that each tri-task of the workflow is assigned a pair of nodes which is a feasible resource allocation for this particular tri-task. $w(j) = \theta$ means that a computational task v_j has been assigned a resource node X_θ. The above-mentioned algorithm for finding feasible resource allocations for workflow W was termed the RA_W algorithm. It was proved in [6] that the RA_W algorithm can always find a feasible resource allocation if it exists, regardless of which pair of precedence-related computational tasks is allocated first. Moreover, finding one such allocation (out of many possible) can be done in polynomial time.

III.2. Scheduling in the GSSIM environment

Different approaches to the problem of grid resource allocation must be carefully studied and tested before they find application in a production environment. However, experiments aimed at evaluating and comparative analysis of these scenarios are often impossible to perform. This is caused by the difficulties in obtaining access to a large-scale infrastructure with its dynamic nature. In addition, carrying out such tests is expensive and time-consuming. Hence, simulations are a common tool for researching various solutions. Also, they facilitate the evaluation of different configurations and management issues together with suitable "what-if" analyses.

The Grid Scheduling Simulator (GSSIM) is a simulation framework developed at the Poznan Supercomputing and Networking Center. It is a comprehensive and advanced simulation tool for distributed computing problems, which allows the users to perform experiments related to grid resource management, and enables experimental studies of various scheduling algorithms. GSSIM focuses on flexible and automated management of a research experiment, including inserting scheduling algorithms into the simulated environment, modeling realistic workloads and adopting real traces, as well as configuring the environment topology, both on the logical and physical level. Simulating a network based on flows with advance reservation functionality allows researchers to examine existing and future network models. GSSIM also allows modeling the topology and performance characteristics of jobs, which is essential for optimizing the execution of complex and demanding real-world applications.

In general, GSSIM can model two generic types of scheduling entities: global and local schedulers. It provides interfaces that allow researchers to insert their specific global scheduling algorithms and local scheduling policies into the simulation environment. A global scheduler is responsible for scheduling jobs to resources distributed among different administrative domains (sites). To this end, it must interact with multiple sites, including retrieving information about resources and submitting jobs. A common example of a global scheduler is the grid broker. Local schedulers are responsible for managing resources that belong to a single site. They retrieve tasks and schedule them to particular subcomponents of a given site. Thus, they correspond to the functionality of popular local scheduling systems, such as LSF, PBS, SGE. Their implementation may imitate the behavior of these well-known systems. On the other hand, they may simulate other, possibly hypothetical, systems and resources. The goal of a local scheduling plugin is either to execute an appropriate task (which has just arrived from the queue) or to put the task into the queue, according to the implemented policy. In this manner, the starting time of each task is defined. Scheduling can be performed depending on specific events that occur in the system. GSSIM is able to handle a wide variety of events, including the arrival of a new task, completion of the previously performed task, periodic events, etc. This allows to apply various scheduling strategies: off-line scheduling for entire sets of incoming jobs, dynamic scheduling based on specific events, and periodic rescheduling. For further details of the GSSIM framework cf. [1] and [4]. GSSIM is complemented by a portal [14], which enables online access to the simulator through a user-friendly experiment editor, workload generator and an experiment repository.

IV. COMPUTATIONAL EXPERIMENTS

In the following two sections we discuss the experiments concerning scheduling in grid environments. Section IV.1 describes an experiment which demonstrates the importance of the resource allocation stage. Section IV.2 concerns the influence of the local scheduling policy on the quality of the obtained schedules. For experimental purposes, the overall workflow completion time (makespan) is considered as the evaluation criterion.

We shall first discuss the parameters of the experiments.

The grid topology considered in the experiments is illustrated in Fig. 1. In order to perform comprehensive and reliable studies, all experiments were carried out many times and the obtained results were averaged. In the first phase, the RA-W algorithm was used to find a feasible resource allocation. To avoid the influence of the choice of the pair of precedence-related computational tasks which were allocated first on the obtained resource allocation (see Section III.1), all test were performed by iterating over all tri-tasks and selecting each of them as the starting one. For each tri-task, tests were repeated 20 times. Hence, the total number of conducted experiments may be calculated as the product of the total number of pairs of precedence-related computational tasks and of the number of repetitions. Thus, the number of performed experiments, which aims at a single configuration evaluation of a particular environment, was equal to 1040 in Section IV.1, and 360 in Section IV.2. The detailed characteristics of the tasks and resources are described in the next two sections.

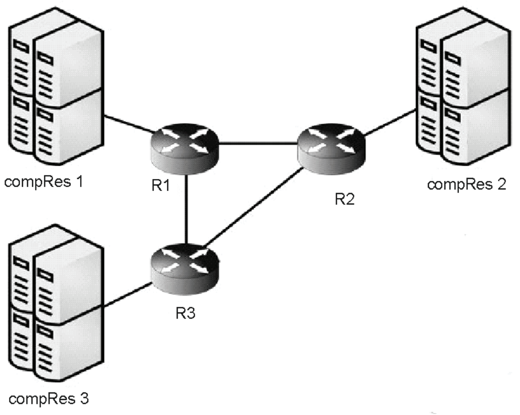

Fig. 1. Grid environment

IV.1. Heuristic approach

According to the proposed model, in order to find a feasible resource allocation for the entire workflow (function *w*), it suffices to select a random feasible resource allocation from the set of all feasible resource allocations for a given tri-task (although for the subsequent tri-tasks this set is limited by previous choices). However, a random choice stands a poor chance of leading to an optimal solution. Hence, we propose a simple modification to the basic version of the RA-W algorithm. This new heuristic algorithm for finding a feasible resource allocation assumes a local minimization of the execution time of the considered tri-task. To this end, from a set of all feasible resource allocations for tri-task $\langle v_i, v_j \rangle$ a pair of resource nodes (X_χ, X_θ) is chosen so as to minimize the sum of the execution time of transmission task (v_i, v_j) between nodes (X_χ, X_θ) and of the execution time of computational task v_j on resource X_θ. The major hypothesis to verify is that deliberate selection of a resource allocation for a given tri-task improves the overall performance with respect to the evaluation criterion.

The structure of the grid environment considered in this experiment is shown in Fig. 1. The characteristics of the two evaluated computational resource configurations are given in Tables 1 and 2, and the parameters of the network resources are presented in Table 3.

Table 1. Computational resource configuration 1

Resource	Number of processors $P_{\kappa\chi}$	Processor speed ϖ_k
compRes1	16	1
compRes2	4	4
compRes3	8	2

Table 2. Computational resource configuration 2

Resource	Number of processors $P_{\kappa\chi}$	Processor speed ϖ_κ
compRes1	16	1
compRes2	4	2
compRes3	8	4

Table 3. Network resource configuration

Link $(\mu,\nu)_\psi$	Bandwidth $\Psi_\psi^{\mu\nu}$
R1_R2	1 Gb/s
R2_R3	1 Gb/s
R1_R3	100 Mb/s
R1_compRes1	1 Gb/s
R2_compRes2	1 Gb/s
R3_compRes3	1 Gb/s

Each computational resource is controlled by a queueing system with a single queue available. To select processing resources for tasks at the local level, the well-known "first-come, first-served" (FCFS) policy was applied, so that the tasks are taken from a queue in the order of their arrival.

Table 4 contains the characteristics of the two types of workflows used in the experiment.

Table 4. Workflow characteristics

Parameter	Number of computational tasks $\lvert V\rvert$		Required number of processors r_i		Min. speed factor ω_i		Number of transmission tasks $\lvert E\rvert$		Required bandwidth B^{ij} [Mb/s]		Size of data file F^{ij} [GB]	
Workflow	W1	W2	W1	W2	W1	W2	W1	W2	W1	W2	W1	W2
AVG	40	40	7	7	1	1	52	52	100	100	107,5	215
STDEV	40	40	5	5	1	1	52	52	100	100	13,4	26,8
MIN	40	40	1	1	1	1	52	52	100	100	80	160
MAX	40	40	16	16	1	1	52	52	100	100	120	240

Table 5. Experimental results for random and heuristic strategies (makespan [s])

Selection	Random				Optimized			
Workflow	W1		W2		W1		W2	
ResConf	Conf1	Conf2	Conf1	Conf2	Conf1	Conf2	Conf1	Conf2
AVG	409 091	407 366	508 366	503 025	40 1258	392 800	484 008	477 283
STDEV	16 391	17 219	22 756	27 211	17 474	20 635	32 480	37 969
MIN	369 600	364 600	446 600	428 600	368 600	351 600	412 600	385 600
MAX	448 600	442 600	572 600	559 600	447 600	436 600	560 600	556 600

As it was mentioned in Section II, the actual execution time of a computational task is calculated as p_i / ϖ_κ, whereas the execution time of a transmission task – as F^{ij} / B^{ij}.

Table 5 shows the results obtained for random and heuristic strategies of resource allocation, for two types of workflows (W1 and W2) and two computational resource configurations (Conf1 and Conf2). This table shows the completion times of the workflows (makespan) – the average, minimum and maximum makespan, as well as the standard deviation. The heuristic strategy significantly outperformed the basic approach – the overall completion time is visibly lower in the case of reasonable selection. This confirms our basic assumption that it is worthwhile to optimize the selection process, even for a single tri-task.

IV.2. Influence of local scheduling policy

Local schedulers manage resources within a single administrative domain, and schedule jobs to local resources. Therefore, one of their main goals is to minimize resource starvation. Generally, it is reasonable to compare different scheduling strategies when the sets of tasks to be scheduled are identical. However, since we find a feasible resource allocation by assuming a simple random selection algorithm and a random selection of the start task, it becomes difficult to obtain the aforementioned result at the local scheduler level. Therefore, this experiment focuses on showing the impact of local policies on the execution start time of both computational and transmission tasks, and thus on the completion time of the entire workflow, rather than on a comparison of scheduling algorithms.

The structure of the grid environment considered in this experiment is shown in Fig. 1. The characteristics of computational and network resources are presented in Tables 6 and 7, respectively.

Table 2. Characteristic of computational resources

Resource	Number of processors $P_{\kappa\chi}$	Processor speed ϖ_κ
compRes1	16	1
compRes2	4	1
compRes3	8	1

Table 3. Characteristic of network resources

Link $(\mu,\nu)_\psi$	Bandwidth $\Psi_\psi^{\mu\nu}$
R1_R2	1 Gb/s
R2_R3	1 Gb/s
R1_R3	100 Mb/s
R1_compRes1	1 Gb/s
R2_compRes2	1 Gb/s
R3_compRes3	1 Gb/s

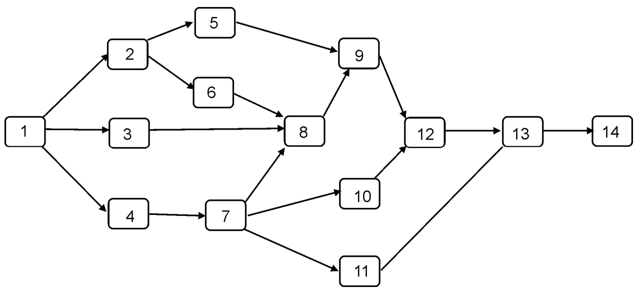

Fig. 2. Workflow topology

Table 4. Characteristics of computational tasks

Computational task v_i	Size p_i	Required number of processors r_i	Min. speed factor ω_i
1	7 200	4	1
2	14 400	8	1
3	14 400	8	1
4	7 200	4	1
5	21 600	12	1
6	18 000	10	1
7	7 200	4	1
8	14 400	8	1
9	7 200	4	1
10	25 200	14	1
11	28 880	16	1
12	14 400	8	1
13	7 200	4	1
14	7 200	4	1

The workflow topology is illustrated in Fig. 2. Table 8 gives the characteristics of the computational tasks. The same parameters were assumed for all transmission tasks, i.e. the size of the data file F^{ij} = 50 GB, and the required bandwidth B^{ij} = 100 Mb/s.

Since we assumed homogeneous resources at each resource provider, the processing time for each task is identical on each computing node. The characteristics of computational tasks as well as the grid topology were analytically determined and adjusted in order to demonstrate the impact of local scheduling strategies on the completion time of the workflow. The experiments were performed using two local scheduling policies: shortest job first (SJF) and longest job first (LJF). At the local level, we performed off-line scheduling by invoking both evaluated strategies every 1000 seconds. Moreover, a feasible resource allocation for the entire workflow (function *w*) was found by random selection of particular feasible resource allocations. Table 9 shows the makespan for both local policies.

Table 5. Results of the experiment for SJF and LJF policies (makespan [s])

Policy	SJF-1000	LJF-1000
AVG	159 885	181 442
STDEV	16 281	8 983
MIN	133 600	171 200
MAX	180 200	201 200

The SJF algorithm turned out to be more effective than the LJF algorithm in terms of the overall completion time. Figures 3 and 4 show examples of schedules obtained with the SFJ and LJF policies. The arrow in Fig. 4 indicates the task (task 6) responsible for the differences in performance between these two algorithms in a significant number of schedules.

Considering the structure of the workflow (cf. Fig. 2), we suspect that task 8 is a possible bottleneck. Its execution depends on the completion of three transmission tasks related to the preceding computational tasks. However, all transmission tasks have to be completed before task 8 can

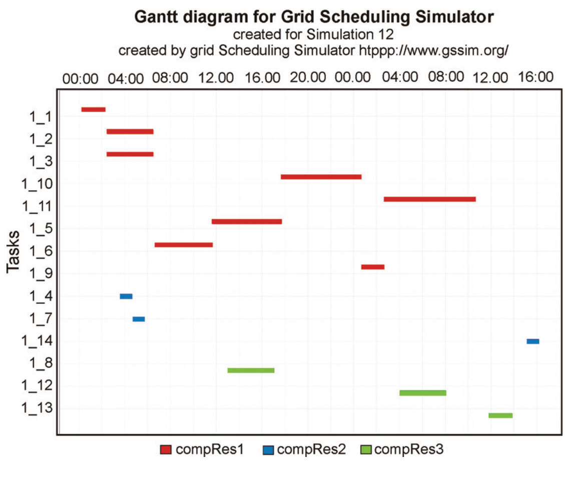

Fig. 3. Schedule obtained with SJF

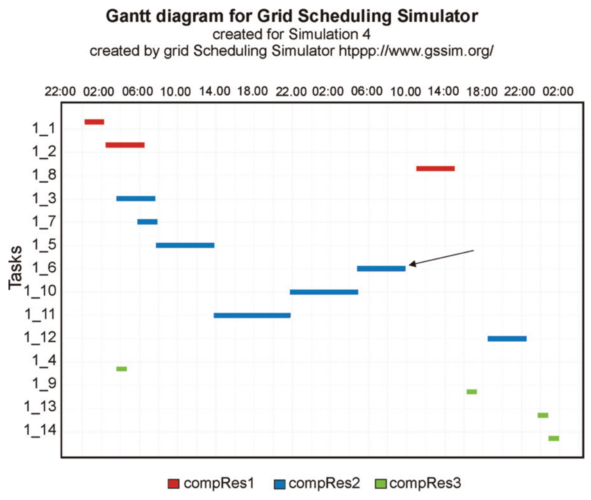

Fig. 4. Schedule obtained with LJF

be started. Therefore, due to the delayed execution of task 6, tasks 8 and 9 are delayed as well, even though remaining precedence constraints have already been satisfied. Transmission tasks can also be considered as possible communication bottlenecks. Large data files are time-consuming and therefore require a large amount of bandwidth in order to be performed efficiently. Hence, such data files should be transmitted first in order to allow the execution of dependent computational tasks. This confirms the importance of an efficient resource allocation in a grid environment, particularly in the case of workflow applications.

V. CONCLUSIONS AND FUTURE RESEARCH

In this paper, the problem of scheduling workflow applications in a grid environment was considered. Grid resources were divided into two types: computational resources and network resources. Accordingly, computational tasks of the workflow as well as transmission tasks were distinguished. The problem can be decomposed into two subproblems: (i) how to find a feasible resource allocation of distributed grid resources for tasks of the workflow in such a way that the resource demands of all tasks (both computational, and transmission) are satisfied, and (ii) how to schedule computational tasks on local resources managed by local grid schedulers. The objective is to minimize the total completion time of the workflow, i.e. the makespan. Computational experiments were performed to justify the importance of the resource allocation stage, as well as examine the influence of the local scheduling policy. The experiments showed that even a small improvement in resource allocation, as compared to random allocation, can result in significantly improved schedules. This confirms our hypothesis about the importance of the resource allocation stage (Section IV.1). As far as local scheduling policies are concerned, the SJF algorithm produced slightly better results than the LJF algorithm. However, the main goal of the second experiment was to demonstrate the influence of the local scheduling policy in the context of the resource allocation stage This was analyzed and discussed in Section IV.2.

In the future, we intend to perform comprehensive computational experiments concerning the problem of scheduling workflows on a grid. In these experiments we hope to show that efficient heuristic (or metaheuristic) algorithms used at the level of global scheduling can significantly improve the quality of the obtained schedules, particularly in the case of workflow applications.

Acknowledgements

This work has been funded by the Polish Ministry of Science and Higher Education as a research project in 2010-2012 under grant No. N N519 579038.

References

[1] S. Bąk, M. Krystek, K. Kurowski, A. Oleksiak, W. Piątek, J. Węglarz, *GSSIM – a Tool for Distributed Computing Experiments*. Scientific Programming 19(4), 231-251 (2011).
[2] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, S. Korranda, *Mapping abstract complex workflows onto Grid environments*. Journal of Grid Computing 1(1), 25-39 (2003)
[3] I. Foster, C. Kesselman, *Computational Grids*, in: I. Foster and C. Kesselman (eds.) *The Grid: Blueprint for a New Computing Infrastructure*, San Francisco: Morgan Kaufmann 15-52 (1999).
[4] K. Kurowski, J. Nabrzyski, A. Oleksiak, J. Węglarz, *GSSIM – Grid Scheduling Simulator*. Computational Methods in Science and Technology 13(2), 121-129 (2007).
[5] K. Kurowski, J. Nabrzyski, A. Oleksiak, J. Węglarz, *Multicriteria approach to two-level hierarchy scheduling in Grids*. Journal of Scheduling 11(5), 371-379 (2008).
[6] M. Mika, G. Waligóra, J. Węglarz, *Modelling and solving grid-resource allocation problem with network resources for workflow applications*. Journal of Scheduling 14(3), 291-306 (2011).
[7] http://www.cs.wisc.edu/condor/
[8] http://www.platform.com/
[9] http://www.pbsgridworks.com/
[10] http://gridengine.sunsource.net/
[11] http://www.eu-egee.org/
[12] http://clusterix.pcz.pl/
[13] http://www.coregrid.net/
[14] http://www.gssim.org

Marek Mika, PhD, graduated from the Poznan University of Technology in Computing Science in July 1994. He obtained his PhD degree in computing science at the Poznan University of Technology in 2001. Since 1994, he has worked at the Institute of Computing Science, in the Laboratory of Operational Research and Artificial Intelligence at the Poznan University of Technology. Since 2001, he has worked as an assistant professor. His main areas of interest are: resource-constrained project scheduling, metaheuristic algorithms, resource management and scheduling in computational grids. He authored or co-authored over 60 scientific publications in international journals, monographs and conference proceedings published by Springer, Elsevier, Kluwer, etc. He was involved in several research projects. The results of his works were presented during over 30 domestic and international scientific conferences and workshops. As a reviewer, he contributes to several international journals in the field of decision science. He is currently working on his habilitation thesis.

Wojciech Piątek received his M.Sc. degree in Computer Science from the Poznan University of Technology in 2010. His master's dissertation concerned the management of computational and network resources on a grid. Since 2008, he has worked in the Applications Department in the Poznan Supercomputing and Networking Center (PSNC) as a system analyst and developer. He participated in the PL-Grid project, and he is currently involved in the CoolEmAll project. He is also one of the developers of the Grid Scheduling Simulator (GSSIM). During the summer of 2010, he was a member of the HPC research group in the Center for Research Computing at the University of Notre Dame. His research activities concern distributed computing, particularly scheduling and resource management in grids, clouds and HPC systems. Additionally, his research interests are focused on *mathematics and algorithms.*

Grzegorz Waligóra, DSc, graduated from the Poznan University of Technology in Computing Science in July 1994. Since 1994, he has worked at the Institute of Computing Science, in the Laboratory of Operational Research and Artificial Intelligence at the Poznan University of Technology. Currently he is an assistant professor. In 2000 he received his PhD degree in Computing Science from the Poznan University of Technology. He published his habilitation thesis on discrete-continuous project scheduling in 2009. His main areas of interest are: project and machine scheduling, discrete-continuous scheduling, combinatorial optimization, metaheuristic algorithms, resource management and scheduling in computational grids. He has authored or co-authored over 70 scientific publications in international journals, monographs and conference proceedings, and presented his results during 40 domestic and international scientific conferences and workshops. He was awarded the Prize of the Foundation for Polish Science for Young Researchers (1999) and the Prime Minister's Award for his PhD Thesis (2000).

Jan Węglarz, professor (PhD 1974, DSc 1977), between 1978 and 1983 he was an associate professor and then professor at the Institute of Computing Science, Poznan University of Technology. He has been a full member of the Polish Academy of Sciences (PAS) since 1998 (a corresponding member since 1991), director of the Institute of Computing Science at the Poznan University of Technology since 1987, director of the Poznan Supercomputing and Networking Center since 1997, president of PAS (Poznan branch) in 2002-2010, president of the Committee for Computer Science of PAS since 2007, founder and chairman of the EURO Working Group on Project Management and Scheduling, principal editor of the Foundations of Computing and Decision Sciences, and a member of several editorial boards, among others: European Journal of Operational Research and International Transactions in Operational Research. He represented Poland in the Board of Representatives of IFORS and in EURO Councils (president of EURO Council in 1997-98). He is a member of several professional and scientific societies, among others: the American Mathematical Society and the Operations Research Society of America. His major research areas include: scheduling (project, machine, production), grid resource management systems, energy-aware computing. He authored and co-authored 15 monographs, 3 textbooks, and over 200 papers in major professional journals and conference proceedings. He is a frequent visitor to major research centers in Europe and in the USA. He is a co-laureate of the State Award (1988) and the EURO Gold Medal (1991), a laureate of the Prize of the Foundation for Polish Science (2000). He received honorary degrees from: Technical University of Szczecin (2001), Academy of Mining and Metallurgy (2002), Technical University of Czestochowa (2005), Poznan University of Technology (2006), Gdansk University of Technology (2008), University of Silesia (2009), and University of Zielona Gora (2009).