

# ON THE CHARACTERISTICS OF GRID WORKFLOWS

Simon Ostermann, Radu Prodan, and Thomas Fahringer

*University of Innsbruck, AT*

simon@dps.uibk.ac.at

radu@dps.uibk.ac.at

tf@dps.uibk.ac.at

Alexandru Iosup and Dick Epema

*Delft University of Technology, NL*

A.iosup@tudelft.nl

D.H.J.Epema@tudelft.nl

**Abstract** Grid computing promises to enable a reliable and easy-to-use computational infrastructure for e-Science. To materialize this promise, grids need to provide full automation from the experiment design to the final result. Often, this automation relies on the execution of workflows, that is, of jobs comprising many inter-related computing and data transfer tasks. While several grid workflow execution tools already exist, not much is known about their workload. This lack of knowledge hampers the development of new workflow scheduling algorithms, and slows the tuning of existing ones. To address this situation, in this work we present an analysis of two workflow-based workload traces from the Austrian Grid. We introduce a method for analyzing such traces, focused on the intrinsic and on the environment-related characteristics of the workflows. Then, we analyze the workflows executed in the Austrian Grid over the last two years. Finally, we identify six categories of workflows based on their intrinsic workflow characteristics. We show that the six categories exhibit distinctive environment-related characteristics, and identify the categories that are difficult to execute for common workflow schedulers.

**Keywords:** grid, workflow execution, workload traces, workflow characteristics, statistic analysis

## 1. Introduction

The grid computing vision aims for a simple useable, dependable, and efficient computing architecture and structure. For this vision to become reality, grids must fully automate the process that starts with experiment design and ends with analysis results. In turn, the full automation necessarily involves the execution of workflows, that is, of jobs with a task graph structure and comprising computing and data transfer tasks [12–13]. While several grid workflow execution engines have recently emerged [17], not much is known about their demand, impacting adversely the evolution of old and new workflow engines. To address this situation, in this work we analyze the workflow-based e-Science applications executed in the Austrian Grid for the past two years.

Currently, there are no publicly available traces of workflow-based grid workloads, that is, of grid workloads that include workflows. This lack of information hampers the testing and the tuning of existing workflow engines, and the study and evolution of new workflow scheduling algorithms. Without proper testing workloads, workflow engines may fail when facing high load or border cases of workload characteristics. Without detailed workload knowledge, tuning lacks focus and leads to under-performing solutions. Without an understanding of real workloads, current research studies use synthetically generated workflows, and are limited in scope and applicability. Moreover, evolution lacks real problems, and may lead to impractical solutions. Understanding the characteristics of existing grid workloads is key to alleviating all these issues. As a first step towards understanding grid workflows, we study two long-term workload traces from the Austrian Grid. The goal of this work is the analysis of the traces and not comparison of their underlying environment. However, the data alone are insufficient: there is a need for new methods to extract and analyze the workflow characteristics from the workload traces. Furthermore, there is a need to identify the class(es) of workflows for which the execution environment yields distinctively poor performance. Our contribution is threefold:

- 1 We propose a method for analyzing the intrinsic and the environment-related characteristics of workflow-based grid workloads;
- 2 We apply the proposed method on two long-term grid workload traces taken from the Austrian Grid;
- 3 Based on the results of the analysis, we identify six classes of workflows with distinct properties, facilitating the identification of the classes which the execution environments can handle the worst to identify improvement possibilities.

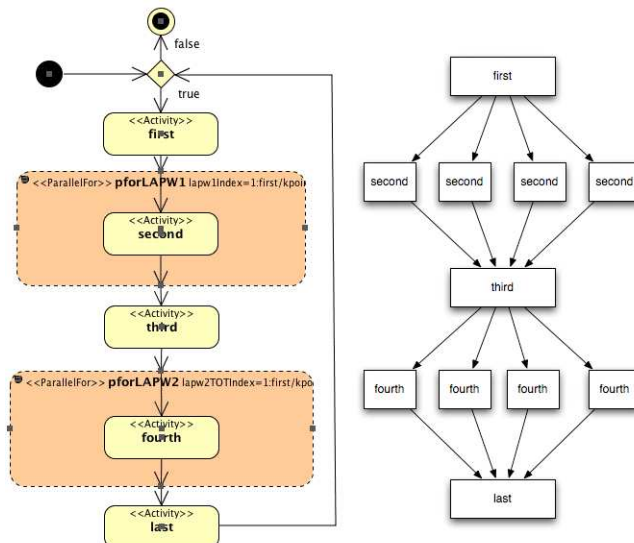


Figure 1. A sample workflow: Wien2k [1]. (left) in AGWL. (right) a possible DAG.

## 2. Background

In this section we present the background information necessary for following this work. We first introduce the workload model used throughout this work. Then, we present DEE and EE2, the workflow engines that executed the workflows investigated in this work.

### 2.1 A Workflow Model

We use for grid workflows the model introduced by Coffman and Graham [12]. Formally, a workflow is a directed graph  $G$  in which nodes are computational tasks, and the directed edges represent communication; we denote the number of nodes and edges by  $N$  and  $E$  respectively. To begin a task, all its predecessor tasks and all communication with that task as the sink must be finished. For simplicity, we consider only directed acyclic graphs (DAGs), that is, there does not exist a loop in the directed graph which may require loop unrolling. We call *root* a node without predecessors; a DAG may have several roots. We further define a *node's level*, derived from breadth-first traversal of the task graph from its roots [2], as the minimal size of a path from the top to this node (in number of edges); the level of a root is 0. Finally, we call the maximum level of a leaf in the graph the *graph level*, which we denote by  $L - 1$ ; we denote by  $L$  the *graph traversal height*. Note that parameter sweeps and other batches of jobs [10] may be considered as degenerate workflows with  $L = 1$ , and that master-worker applications can be seen as workflows with  $L = 3$  (including a final task to assemble the worker results).

Figure 1 (left) shows a sample workflow from ASKALON [4], defined by the user in the Abstract Grid Workflow Language (AGWL) [5]. Figure 1 (right) depicts an instance of this example that has one iteration of the outermost while loop. The number of nodes  $N$  is 11, the number of edges  $E$  is 16, and the graph traversal height  $L$  is 5.

## 2.2 The ASKALON Workflow Engine

The ASKALON grid middleware [4] can execute via its workflow execution engine workflows specified in AGWL. While execution, this abstract specification is instantiated, that is, the tasks are annotated with details concerning the used resources. ASKALON's workflow execution engine features a fine grain event system which is implemented as WSRF service and allows event-forwarding even through NAT or firewalls. An overview of this and other grid workflow systems can be found in the taxonomy of workflowsystems [17].

In the past two years, the ASKALON workflow execution engine evolved in two major steps. The first version, DEE [3], focused on functionality. DEE's primary shortcomings were the internal loop unrolling from the workflow specification, and the complete scheduling at the start of the execution. To improve on scalability and on adaptability to highly dynamic grid environments, the second generation engine EE2 was developed [16]. The EE2 uses internally a structure that is kept close to the AGWL specification and better scales for the execution of large workflows. Each job that is ready for execution will dynamically be send to the best available grid site at this moment.

## 3. A Method for Workflow-based Grid Workloads Analysis

In this section we introduce a method for analyzing workflow-based grid workloads. The goal of our analysis is to establish the main characteristics of the workload such that building a workflow-based grid workload model is greatly facilitated. For our method to be applicable, the analyzed traces need to be long-term (i.e. at least a month) and to provide sufficient statistical confidence (i.e., to include at least several hundreds of workflows each). From a technical point of view, we extract for each workflow characteristic a comprehensive set of statistical properties (i.e. min, max, average, std. deviation and quantiles). We also compute the empirical distribution of the characteristics. For an overview of these statistical tools we refer to Jain's classical text [11].

Our method divides the characteristics into two classes, workflow-intrinsic and environment-related. We first analyze the intrinsic workflow characteristics in Section 3.1, which allows us to identify the workflow classes that may have different environment-related characteristics. Then, we analyze the environment-related characteristics for the complete workload data and per class in Section 3.2.

### 3.1 Intrinsic Workflow Characteristics

The intrinsic workflow characteristics refer to the size and the structure of the workflows, and to their arrival pattern. We assume that users are not influenced by the system properties (e.g. size) when defining their workload, and that the submission of the workflows is independent from the state of the system (though the submission of the tasks to the grid by the workflow engine may not be).

To characterize the workflow size and graph structure, we employ the following characteristics (listed in the chronological order of their analysis within our method):

**Number of nodes (N), Number of edges (E), Graph level (L) ;**

**Branching Factor (BF)** defined as the ratio between the number of edges  $E$  and the number of nodes  $N$ . The branching factor may have a high impact on the graph execution time: the higher the branching factor, the higher the probability that a task's execution is delayed due to waiting for its predecessors.

**Work Size** defined as task runtime of a task on a base platform. To compute the task work size we face the problem of data coming from a heterogeneous environment: the same task may take different runtime when executed by different resources. To compute the work size, we normalize the task runtime logged in the workload trace with the ratio between the performance of the resource on which the task is executed and that of a base resource. Following the example of CERN's WLCG, at over 50,000 computing resources the largest grid that publishes size information, we express the resource performance in SPECInt2000 values. SPECInt2000 is a collection of twelve benchmarks representative for industrial and scientific applications [7]. The choice of using the SPECInt2000 values is based on the implicit assumption that this benchmark is representative for the applications executed in the studied workload trace.

**Variability of the work size inside a workflow (WSV)** defined as the ratio between the runtime of the longest and of the shortest task. The higher the variability, the more difficult it is for a task or workflow runtime predictor [18] to operate, leading to potentially low performance for the workflow scheduler.

**Sequential execution path** defined for a workflow as the sum of the work sizes of its tasks;

**Critical execution path** defined for the longest execution path in the workflow graph. Any delay of a task on this path will result in a delay of the total executions end. [2].

<i>Trace</i>	<i>Source</i>	<i>Duration</i>	<i>Number of WFs</i>	<i>Number of Tasks</i>	<i>CPUDays.</i>
T1	DEE	09/06-10/07	4,113	122k	152
T2	EE2	05/07-11/07	1,030	46k	41

Table 1. Workflow-based traces analyzed for this work. (WF stands for workflows.)

## 3.2 Environment-Related Workflow Characteristics

The environment-related workflow characteristics are time-related (e.g., wait, run, and makespan), scheduler-related (e.g., makespan vs. critical path), and failure-related (e.g., amount of failures).

We employ the following performance metrics:

**Makespan** defined for a workflow as the time elapsed between the workflow’s entering and exiting the system.

**Speedup (S)** defined as the ratio between a workflow’s makespan and its sequential execution path size.

**Normalized Schedule Length (NSL)** [13], defined as the ratio between a workflow’s makespan and its critical path size.

**Success Rate (SR)** defined for a workflow as the percentage of tasks that finished correctly from the workflow tasks.

## 4. The Results

In this section we describe the long-term trace data collected from the Austrian Grid, and the results obtained when applying the method described in the previous section to these data.

### 4.1 The Workload Traces

In this work we use two traces collected from the Austrian Grid over a period of more than one year. The T1 (T2) trace was collected from the system using the DEE (the EE2) as workflow engine (see Section 2.2). Table 1 summarizes the characteristics of the traces used in this work. Each of the traces contains information about more than a thousand workflows, satisfying our analysis method’s input requirements.

The collected traces consist mostly of workflow test runs, albeit often of production workflows, done by the developers of the system in order to find bugs and drive development. Thus, the results presented in this section should be regarded as guiding, but not definitive, in establishing the characteristics of the grid workflows.

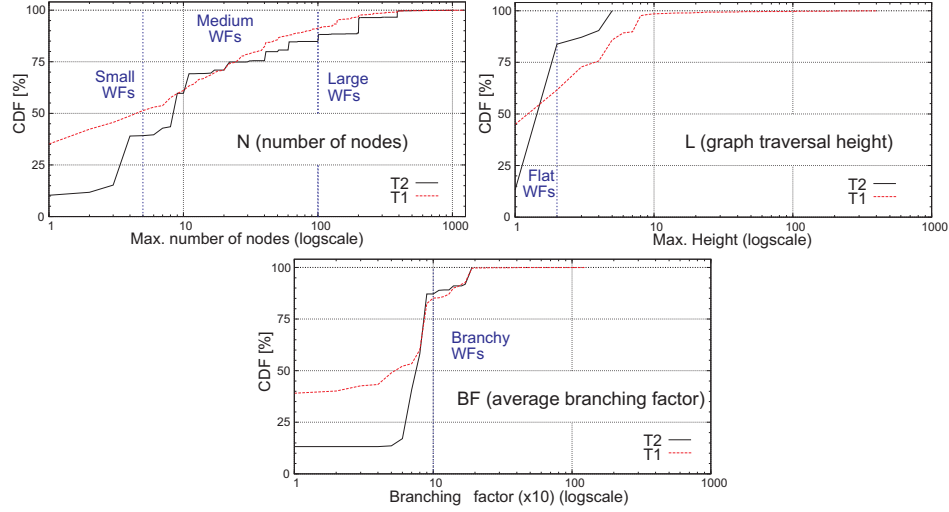


Figure 2. The workflow structure for T1 and T2. (top left) number of nodes. (top right) graph traversal height. (bottom) average branching factor.

## 4.2 The Intrinsic Workflow Characteristics

Figure 2<sup>1</sup> shows the graph size and structure of the workflows from both studied traces. The average number of tasks per workflow is  $30 \pm 70^2$  for T1 and  $44 \pm 91$  for T2. The fact that the average is much higher than third quartile (25 and 31 for T1 and T2, respectively) indicates a skewed distribution, confirmed by Figure 2(bottom). For both traces, 75% of the workflows have fewer than 40 tasks, and 95% of the workflows have fewer than 200 tasks. The average branching factor is  $0.64 \pm 0.67$  for T1 and  $0.86 \pm 0.46$  for T2. This indicates that users prefer to submit loosely-coupled tasks, that is, tasks that do not depend on many previous results. Thus, the properties of  $N$  and those of  $E$  are very similar (we have also validated this finding separately). The graph level is  $3.73 \pm 14.04$  for T1 and  $2.25 \pm 1.0$  for T2, indicating that the new engine is mostly used for graphs with little depth. For T2, slightly over 80% of the workflows have at most two levels.

Based on the results for the graph size and structure and experience from [10], we define the workflow classes summarized in Table 2; mixed classes can also be formed. The Small, Medium, and Large classes refer to the workflow size. The Branchy class contains workflows with more edges than usual. The

<sup>1</sup>All CDF graphs in this paper are discrete as they map real values to their integral part before the accumulation.

<sup>2</sup>We use throughout this work the notation  $\mu \pm \sigma$  to denote a set of values with the average  $\mu$  and the standard deviation  $\sigma$ . Note that in some cases the values below or equal to zero are not meaningful, e.g., for the number of tasks in the workflow.

<i>Class</i>	<i>N</i>	<i>L</i>	<i>BF</i>
Small	<5	-	-
Medium	5–100	-	-
Large	>100	-	-
Branchy	-	-	$\geq 1.0$
Flat	-	$\leq 2$	-

Table 2. Classes of workflows based on their size and structure properties.

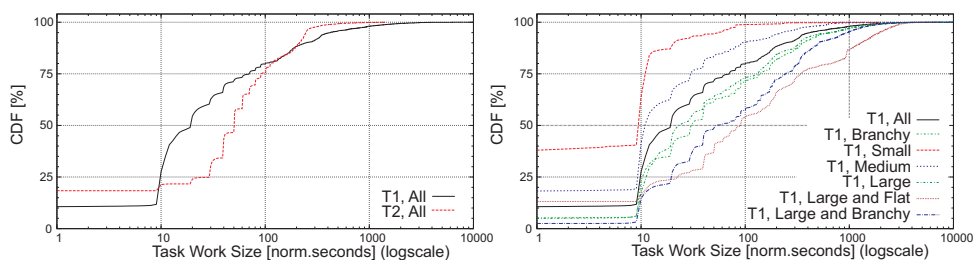


Figure 3. The distribution of the work sizes of the workflow tasks. (left) overall. (right) per workflow class.

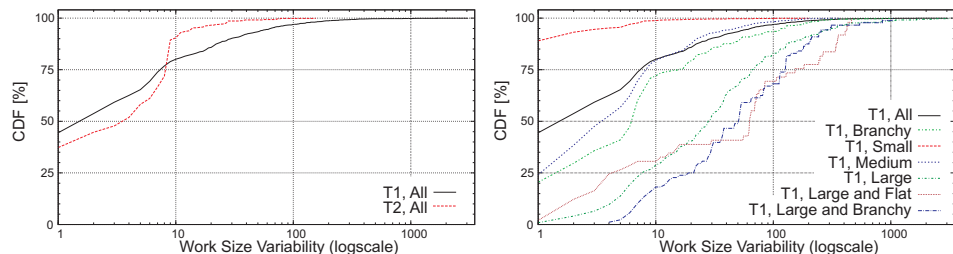


Figure 4. The distribution of the work size variability of the workflow tasks. (left) overall. (right) per workflow class.

Flat class contains workflows with at most two levels. Throughout the rest of this work we focus on the following classes: Small/Medium/Large(all the size-related classes), Branchy, Large and Flat, and Large and Branchy.

Figure 3 shows the CDF of the workflow tasks' Work Size, in normalized seconds (the runtime on a machine with a speed of 1000 SPECInt2000). While over 75% of the tasks take less than 100 seconds, the other 25% can take up to 4 hours (half an hour) for T1 (T2). Around 25% of the large and flat tasks took over 350 seconds while only 2% of the small tasks takes longer than 80 seconds. The medium runs have an overall lower task work size compared to the total (all), while the large a higher size.

Figure 4 shows the CDF of the workflow tasks' Work Size variability (see Section 3.1). More than 75% of the workflows have a Work Size variability below an order of magnitude of the task runtime variability. The maximum

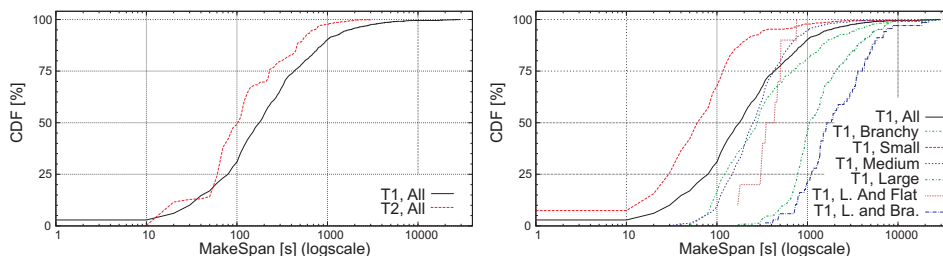


Figure 5. The distribution of the makespan of the workflows. (left) overall. (right) per workflow class.

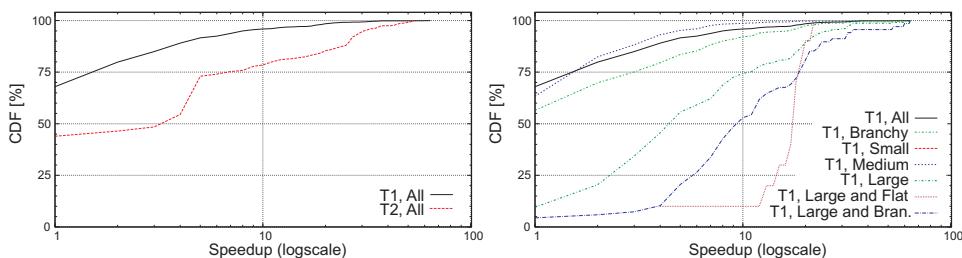


Figure 6. The distribution of the speed-up of the workflows. (left) overall. (right) per wf class.

variability is over 3000 (150) for T1 (T2). Around 95% of the small workflows have a variability smaller than 5 while approximately 20% of all the large workflows have a variability lower than 5.

### 4.3 The Environment-Related Workflow Characteristics

Figure 5 shows the CDF of the workflow makespan. Less than 5% of the T1 workflows take more than one hour; none of the T2 workflows reach one hour. Over 50% of the workflows take less than 4 minutes. As expected large workflows have a higher makespan than short ones.

Figure 6 shows the CDF of the workflow speedup. The average speed-up is  $2.54 \pm 4.92$  for T1 and  $8.21 \pm 11.11$  for T2; the median is 1.15 for T1 and 4.61 for T2. The first quartile value is 0.41 for T1 and 1.70 for T2. This means that for T1 the benefit of executing workflows is in general reduced; also for T1, 25% of the workflows are slowed down by being executed as workflows as opposed to being executed sequentially on a single processor. For T2, the benefits of grid workflow execution are much more visible, with quantile Q1, the median, and the average well above 1. Over 75% of the large and flat workflows were able to achieve a speed up higher than 15. About 50% of the large workflow runs gained a speedup of 9 and higher.

Figure 7 shows the CDF of the workflow normalized schedule length. The NSL values confirm the findings regarding the speedup (given the high correlation between the number of nodes and the number of edges). The average

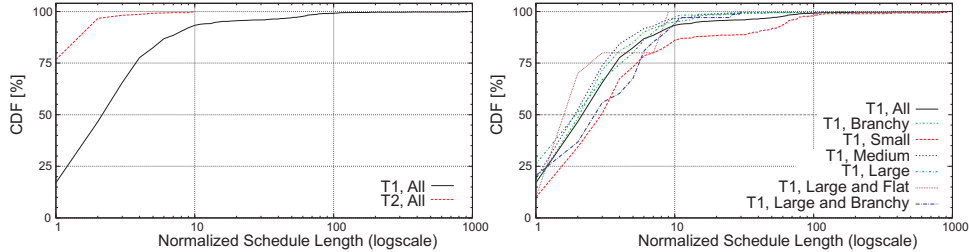


Figure 7. The distribution of the normalized schedule length of the workflows. (*left*) overall. (*right*) per workflow class.

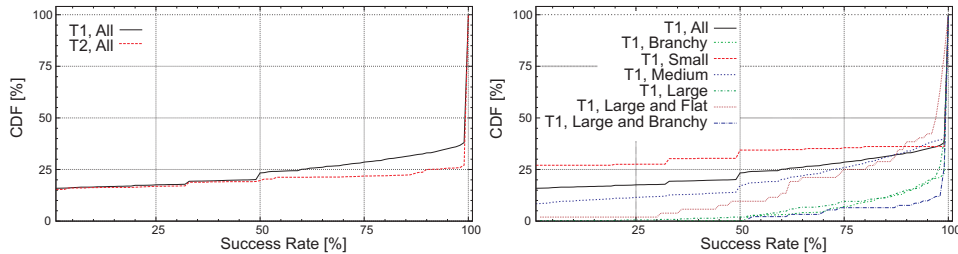


Figure 8. The distribution of the success rate of the workflow tasks. (*left*) overall. (*right*) per workflow class.

NSL is 1.72 for T1 and 1.58 for T2. This shows that the system used for T2 has less overheads in total and will be able to execute workflows on the grid more efficient. In the per class view, the large and flat workflows show special behavior as there where runs with short tasks where the overheads account more and longer runs where the total time could compensate the overheads.

Figure 8 shows the CDF of the workflow success rate. The percentage of workflows with a higher success rate then 98% are 65%, respectively 75% for trace T1 and T2. The large and branchy workflows for T1 have reached a success rate of 97% for even 89% of their runs while the large and flat only get more then 95% success rate for 60% of the traced workflows.

## 5. Related Work

There exists a lot work related to workload analysis and modeling for testing, tuning, and resource management design purposes [6, 13, 15, 14, 9, 8, 10].

Following the seminal work of Feitelson [6], several models for workloads of large computing environments have emerged [15, 14]; a comparative analysis of the characteristics of workloads from production grid environments is first presented in [8]. However, none of these research results is targeting workflows. Iosup et al. [10] present an investigation of the properties of batches of jobs (a degenerate case of workflows that is popular in today's grids) in grids.

Synthetic workloads have been employed to test functionality in real and simulated environments; most of workflow engines have been tested with such workloads. Workflows from the Standard Task Graphs online archive have been reportedly used to test the ability of a multi-cluster grid to execute workflow-based grid workloads [9].

Closest to our work, Kwok and Ahmad propose four models of workflows used for testing in simulation traditional workflow scheduling algorithms [13]. From these models, one model consists of pathological cases identified by scientists for the purpose of stressing their algorithms, two are synthetic models, and only one is extracted from real workloads. The latter models the flow of two parallel applications. Our analysis presents results that complement this latter model with a much broader application selection base (the real workload traces), and for which we can show the environment-related characteristics of executing these workloads in a real heterogeneous environment (including failures). We also add the workflow task work size (and, notably, its variability) as a parameter to model.

## **6. Conclusion and Ongoing Work**

Realistic data concerning the characteristics of workflow-based grid workloads is key to the adoption and the evolution of grids, but is not readily available to scientists. To address this issue, in this work we present the characteristics of two long-term traces from the Austrian Grid, a grid environment in which workflows are common.

We introduce a method for analyzing such traces, then apply the method to the two Austrian Grid traces. The method identifies two broad classes of workflow characteristics, intrinsic and environment-related. Based on the observed values for the former, we devise six classes of workflows with distinct properties. The analysis of environment-related characteristics reveals that from the six classes several can be considered classes of "problem-workflows", which exhibit one or all of high variability of the work size of their tasks, high makespan, poor scalability, and higher than normal failure rate. Overall, we find that the workflow speedup is highly dependent on the system used for execution, and that the current task success rate requires more fault tolerance mechanisms, especially for large workflows.

We plan to extend our work with the analysis of other traces. In particular, we hope to find traces that include mostly production workflows submitted by real users. Based on these traces, we will be able to design a model for workflow-based grid workloads. We conclude by addressing the whole grid community with a request for making available their (workflow-based) grid workload traces to other researchers.

## Availability

The results presented in this work are publicly available as part of the Grid Workloads Archive at: <http://gwa.ewi.tudelft.nl/>

## Acknowledgements

This work is supported by the European Union through IST-004265 and IST-2002-004265 CoreGRID and partially carried out in the context of Virtual Laboratory for e-Science project ([www.v1-e.nl](http://www.v1-e.nl)), supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science, and which is part of the ICT innovation program Affairs. This work is co-funded by the European Commission through the EGEE-II project INFISO-RI-031688.

## References

- [1] P. Blaha, K. Schwarz, and J. Luitz. WIEN2k, a full potential linearized augmented plane wave package for calculating crystal properties. Austria 1999. ISBN 3-9501031-1-2.
- [2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 1989.
- [3] R. Duan, R. Prodan, and T. Fahringer. Dee: A distributed fault tolerant workflow enactment engine for grid computing. In *HPCC*, volume 3726 of *LNCS*, pages 704–716. Springer-Verlag, 2005.
- [4] T. Fahringer, A. Jugravu, S. Pillana, R. Prodan, C. S. Jr., and H. L. Truong. ASKALON: a tool set for cluster and grid computing. *CP&E*, 17(2-4):143–169, 2005.
- [5] T. Fahringer, J. Qin, and S. Hainzer. Specification of grid workflow applications with agwl: an abstract grid workflow language. In *CCGrid*, pages 676–685. IEEE CS, 2005.
- [6] D. G. Feitelson and L. Rudolph. Metrics and benchmarking for parallel job scheduling. In *JSSPP*, volume 1459 of *LNCS*, pages 1–24. Springer, 1998.
- [7] J. L. Henning. Spec cpu2000: Measuring cpu performance in the new millennium. *IEEE Computer*, 33(7):28–35, 2000.
- [8] A. Iosup, C. Dumitrescu, D. Epema, H. Li, and L. Wolters. How are real grids used? the analysis of four grid traces and its implications. In *GRID*, pages 262–269. IEEE CS, 2006.
- [9] A. Iosup and D. H. J. Epema. Grenchmark: A framework for analyzing, testing, and comparing grids. In *CCGrid*, pages 313–320. IEEE CS, 2006.
- [10] A. Iosup, M. Jan, O. Sonmez, and D. Epema. The characteristics and performance of groups of jobs in grids. In *Euro-Par*, LNCS. Springer-Verlag, August 2007.
- [11] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. May 1991.
- [12] E. G. C. Jr. and R. L. Graham. Optimal scheduling for two-processor systems. *Acta Inf.*, 1972.
- [13] Y.-K. Kwok and I. Ahmad. Benchmarking and comparison of the task graph scheduling algorithms. *J. PDC*, 59(3):381–422, 1999.
- [14] H. Li, D. L. Groep, and L. Wolters. Workload characteristics of a multi-cluster supercomputer. In *JSSPP*, volume 3277 of *LNCS*, pages 176–193. Springer-Verlag, 2004.
- [15] U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *J. PDC*, 63(11):1105–1122, 2003.
- [16] K. Plankensteiner. A scalable execution engine for scientific grid workflows. U.Innsbruck, Master Thesis, 2007 (in progress).
- [17] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *ACM SIGMOD Rec.*, 34(3):44–49, 2005.
- [18] Farrukh Nadeem, Radu Prodan, and Thomas Fahringer. Optimizing Performance of Automatic Training Phase for Application Performance Prediction in the Grid. In *HPCC*, pages 309–321, 2007.