Óbudai Egyetem

Thesis booklet

Dynamic Execution of Scientific Workflows

by

Eszter Kail

Supervisors:

Miklós Kozlovszky, Péter Kacsuk

Doctoral School of Applied Informatics

Budapest, 2016

1 Background of the Research

The increase of the computational capacity and also the widespread usage of computation as a service enabled complex scientific experiments conducted in laboratories to be transformed to in silico experiments executed on local and remote resources. In general these in silico experiments aim to test a hypothesis, to derive a summary, to search for patterns or simply to analyze the mutual effects of the various conditions. Scientific workflows are widely accepted tools in almost every research field (physics, astronomy, biology, earthquake science, etc.) to describe and to simplify the abstraction and to orchestrate the execution of these complex scientific experiments.

A scientific workflow is composed of computational steps that are executed in sequential order or parallel wise determined by some kind of dependency factors. We call these computational steps tasks or jobs, which can be data intensive and complex computations. A task may have input and output ports where the input ports consume data and the output ports produce data. Data produced by an output port is forwarded through outgoing edges to the input ports of subsequent tasks. Mostly we differentiate data flow or control flow oriented scientific workflows. While in the former one the data dependency determines the real execution path of the individual computational steps and data movement path, in the latter one there is an explicit task or job precedence defined.

Scientific workflows are in general data and compute intensive thus they usually require parallel and distributed High Performance Computing Infrastructures (HPC), such as clusters, grids, supercomputers and clouds to be executed. These infrastructures consist of numerous and heterogeneous resources. To hide the complexity of the underlying low-level, heterogeneous architecture Scientific Workfow Managements Systems (SWfMS) have emerged in the past two decades. SWfMs tend to manage the execution-specific hardware types, technologies and protocols whilst providing user-friendly, convenient interfaces to the various user types with different knowledge about the technical details. However, this user-friendly management system hides a complex and thus a continuously changing and an error prone environment for workflow execution.

As a consequence, an environment that is able to change or adapt dynamically should be provided. It means that the Scientific Workflow Management System should provide means to adapt to the new environmental conditions, to recover from failures, to provide alternative executions and to guarantee successful termination of the workflow instances with a probability of p and lastly, but not finally to enable optimization support according to various needs such as optimal time and energy usage.

We differentiated three different aspects of dynamism: Fault tolerance, which is the ability to continue the execution of the workflow in the presence of failures; Optimization, which enables optimized executions according to given parameters (i.e.: cost, time, resource usage, power,...); And intervention and adaptive execution, which enables the user, the scientist or the administrator to interfere with workflow execution during runtime and even that the system adaptively reacts to the unexpected situations.

The present dissertation deals with two of the above mentioned research areas: the fault tolerance and the adaptive and user-steered execution.

1.1 Workflow Structure Analysis

Information obtained from the workflow structure is often involved in research areas connecting to the various phases of of the workflow life cycle (WFLC) (Bahsi 2008) (Gil et al. 2007) (Deelman and Gil 2006) (Ludäscher et al. 2009). One of the most frequently used aspect of workflow structure analysis is makespan estimation. It is a widely accepted method that the tasks constituting the workflow are divided into levels based on the data dependencies between them so that tasks assigned to the same level are independent from each other. Then, for each level, its execution time (which is equal to the time required for the execution of the tasks in the level) can be calculated considering the overall runtime of the tasks of that level (Pietri et al. 2014).

Another important aspect of workflow structure investigation is workflow similarity research. It is a very urgent and relevant topic, because workflow reusability and sharing among the scientists' community has been widely adopted and workflow repositories increase in size dramatically. Apart from workflow sharing and retrieval, the design of new workflows is a critical problem to users of workflow systems (Krinke 2001). It is both time-consuming and error-prone, as there is a great diversity of choices regarding services, parameters, and their interconnections. It requires the researcher to have specific knowledge in both his research area and in the use of the workflow system.

The authors in (Starlinger, Cohen-Boulakia, et al. 2014) divided the whole workflow comparison process into two distinct levels: the level of single modules and the level of the whole workflow. First they carry out a comparison between the task-pairs and thereafter a topological comparison is applied. The existing solutions can be classified as either a structure agnostic, i.e., based only on the sets of modules present in two workflows, or a structure based approach (Starlinger, Brancotte, et al. 2014). The latter group makes similarity research on substructures of workflows, such as maximum common subgraphs (Krinke 2001), or using the full structure of the compared workflows (Xiang and Madey 2007).

In scheduling problems workflow structure investigations are also a popular form to optimize resource mapping problems. Several papers introduce slack time (we use this term as flexibility parameter in our work) and prove that this is an effective metric to be used to adjust the robustness and it can be derived from the workflow structure (Shi, Jeannot, and Dongarra 2006), (Sakellariou and Zhao 2004). Another DAG based approach is presented in (Poola et al. 2014), where the workflow is decomposed into smaller groups of tasks, into Partial Critical Paths (PCP), which consist of the nodes that share high dependency between them, and for those the slack time is minimal. They declared that PCPs of a workflow are mutually exclusive, thus a task can belong to only one PCP.

Fault tolerance is most generally based on external conditions for example on failure statistics about components of execution environments or network components and provenance data from historical executions. However, the scientific workflow itself alone involves valuable features that can be exploited not only in workflow scheduling, resource allocation, but also in fault tolerance and optimization techniques.

The first thesis group deals with structure analysis of scientific workflows.

1.2 Provenance Based Checkpointing Method Based on Workflow Structure Analysis

Concerning dynamic workflow execution fault tolerance is a long standing issue and checkpointing is the most widely used method to achieve fault tolerant behavior.

The checkpoint scheme consists of saving intermediate states of the task in a reliable storage and, upon detection of a fault, restoring the last consistent state. Hence, checkpointing enables to reduce the time to recover from a fault, while minimizing loss of the processing time.

Checkpointing techniques can be investigated from numerous perspectives. Based on the storage where the states are stored we can differentiate lightweight and workflow level checkpointing (Oliner et al. 2005).

According to the level, where the checkpointing occurs we differentiate: application level checkpointing, library level checkpointing and system level checkpointing methods (Garg and Singh 2011), (Jhawar, Piuri, and Santambrogio 2013).

Checkpointing schemes can also be categorized to be full or incremental ones (Agarwal et al. 2004), (Palaniswamy and Wilsey 1993) or even to be coordinated and uncoordinated methods (Meroufel and Belalem 2014).

The efficiency of the used checkpointing mechanism is strongly dependent on the length of the checkpointing interval. Frequent checkpointing may increase the overhead, while rarely made checkpoints may lead to loss of computation. Hence, the decision about the size of the checkpointing interval and the checkpointing technique is a complicated task and should be based upon knowledge specific to the application as well as the system. Therefore, various types of checkpointing optimization have been considered by the researchers.

Young in (Young 1974) has defined the formula for the optimum periodic checkpoint interval concerning to the total execution time, which is based on the checkpointing cost and the mean time between failures (MTBF) with the assumption that failure intervals follow an exponential distribution. Di et al. in (Di et al. 2013) has also derived a formula to compute the optimal number of checkpoints for jobs executed in the cloud. His formula is generic in a sense that it does not use any assumption on the failure probability distribution.

Static optimal checkpointing is often investigated with different conditions (Kwak and Yang 2012), (Nakagawa, Fukumoto, and Ishii 2003).

The drawback of these static solutions lies in the fact that the checkpointing cost can change during the execution if the memory footprint of the job changes, network issues arise or when the failure distribution changes. Thus static intervals may not lead to the optimal solution. By dynamically assigning checkpoint frequency we can eliminate unnecessary checkpoints or where the danger of a failure is considered to be severe extra state savings can be introduced.

To address this problem also adaptive checkpointing schemes have been developed in several papers (Li and Chen n.d.), (Di et al. 2013), (Lidya et al. 2010).

In this work the determination of the checkpointing interval, besides some failure statistics is primarily based on workflow characteristics which is a key difference from existing solutions. To the best of our knowledge our work is unique in this aspect. We demonstrate that we can still get good insight into the number of checkpoints during a job execution in order to achieve a desired level of performance with minimum overhead of the used fault tolerant technique.

Thus we were looking for the answer for the question: whether the results of the workflow structure analysis can be integrated into a checkpointing method in order to decrease the overall cost of the checkpointing.

1.3 Provenance Based Runtime Manipulation

In the past decay a lot of Scientific Workflow Management Systems have been developed that were designed to execute scientific workflows. SWfMSs are mostly bounded to one or more scientific discipline, thus they have their own scientific community and they all have their own language for workflow definition like AGWL (Fahringer, Qin, and Hainzer

2005), GWENDIA (Montagnat et al. 2009), gUSE (Peter Kacsuk 2011), SCUFL (Turi et al. 2007) and Triana Taskgraph (Taylor et al. 2003).

Because of the different requirements that were addressed by the various scientific communities, it is widely acknowledged that the creation of a single standard language for all users of scientific workflow systems is a difficult undertaking that will probably not succeed in being adopted by all communities given the heterogeneous nature of their fields and problems to solve.

The SHIWA project (2010-2012) (Team 2011) was targeted to promote interoperability between different workflow systems by applying both coarse- and fine-grained strategies (Terstyanszky et al. 2014), (Plankensteiner, Prodan, et al. 2013). The fine-grained approach deals with language interoperability by defining and Interoperable Workflow Intermediate Representation (IWIR) language (Plankensteiner, Montagnat, and Prodan 2011) for translating workflows (ASKALON, P-Grade, MOTEUR and Triana) from one DCI to another, thus creating a cross-compiler.

However, monitoring from the scientist's perspective is also very important, moreover, data analysis and dynamic intervention is also an emerging need concerning nowadays scientific workflows (Ailamaki 2011). Due to their exploratory nature they need control and intervention from the scientist to conserve energy and time.

There are several systems that support dynamic intervention such as stopping, or re-executing jobs or even the whole workflow but there is an increasing need to have more sophisticated manipulation possibilities. Vahi et al. (Vahi et al. 2012) introduced Stampede, a monitoring infrastructure that was integrated in Pegasus and Triana and which main target was to provide generic real-time monitoring across multiple SWfMSs. The results proved that Stampede was able to monitor workflow executions across heterogeneous SWfMSs but it required the scientists to follow the execution from a workstation. This solution may be tiring concerning long-term executions. To tackle this, it is possible to pre-program triggers, such as proposed by Missier et al. in (Missier et al. 2010), to check for problems in the workflow and to alert the scientist. In an other paper by Pintas et al. (Pintas et al. 2013) worked out sciLightning, a system that is able to notify the scientist upon completion of certain, predefined events. In their work (Dias et al. 2011) authors managed to implement dynamic parameter sweep workflow execution where the user has the possibility to interfere with the execution and change the parameter of some filtering criteria without stopping the workflow.

Also execution performance analysis during runtime is already integrated in several solutions (Lee, Paton, et al. 2009), (Lee, Sakellariou, et al. 2007), (Oliveira et al. 2014).

Concerning the volume of provenance data generated at runtime another challenging research area is provenance data analysis concerning runtime analysis.

However, most of the existing solutions for dynamism provide limited range of changes, that have to be scheduled a-priori and they do not solve on-the-fly modification of parameter sets, data sets or the model itself. On the other hand adapting the workflow execution to runtime provenance data analyses still remained a challenge.

2 Motivation

The following subsections summarize the motivation of our research which was conducted during the past few years.

2.1 Workflow structure and fault tolerance

The different scientists' communities have developed their own SWfMSs, with divergent representational capabilities, and different dynamic support. Although the workflow description language differs from SWfMS to SWfMS according to their scientific research and needs, it is widely acknowledged that Directed Acyclic Graphs (DAG) serve as a top-level abstraction representation tool. Thus, a scientific workflow can be represented by a graph $G(V, \vec{E})$, where the nodes represent the computational tasks and the directed edges denote the data dependency between them. Concerning graphs a wide range of scientific results have been achieved in order to provide to other scientific disciplines with a simple but easily analyzable model. Also in the context of scientific workflows it is widely accepted tool to analyze problems with the help of graphs in scheduling, workflow similarity analysis and also in workflow estimation problems. However, dynamic execution of scientific workflows is most generally based on external conditions for example on failure statistics about components of execution environments or network elements and provenance data from historical executions. Despite this fact, we think that the structure of the graph representing the scientific workflow holds valuable information that can be exploited also in fault tolerance issues.

The first two thesis groups addresses the following questions to answer:

How much information can be obtained from the structure of the scientific workflows to adjust fault tolerance parameters and to estimate the consequences of a failure occurring during one task concerning the total makespan of the workflow execution?
How can this information be built in a proactive fault tolerance method, in checkpointing?

2.2 Adaptive and user-steered execution

From the scientists' perspective workflow execution is like black boxes. The user submits the workflow and at the end he gets a notification about successful termination or failed execution. Concerning long executions due to the complex nature of scientific workflows it may not be sufficient. Moreover due to the exploratory nature of the scientific workflows the scientist or the user may intend to interfere with the execution and based on monitoring or debugging capabilities to carry out a modified execution on the workflow.

In the third thesis group we were looking for the answers for the questions:

How can scientists be supported to interfere with the workflow execution? How can provenance based user-steering or adaptive execution be realized?

3 Objectives

Motivated by the problems outlined in the previous section the objectives of this thesis can be split into two major parts. The first and second thesis groups deal with problems connected to fault tolerance and the third thesis group concerns with adaptive and user steered workflow execution.

3.1 Workflow structure and fault tolerance

My primary aim was to investigate the effects of individual failures concerning to the actually used fault tolerance method, and then to determine the sensitivity index in order to adjust fault tolerant parameters resulting in a more efficient fault tolerance mechanism and thus in a more efficient execution. In the first thesis group I introduce the flexibility zone (I_i) of a task T_i concerning a certain time delay d, and based on this definition I formulate the sensitivity index (SI) of a scientific workflow model, which gives information on the connectivity property of the workflow and the sensitivity of the workflow structure concerning individual failures arising during the execution. I also introduce the time sensitivity (TS) of a workflow model, which gives information about how sensitive a workflow is concerning the makespan due to a failure. According to the time sensitivity (TS) parameter I give an upper and lower limit for the sensitivity index, and based on the sensitivity index I give a taxonomy of scientific workflows.

3.2 Adjusting the checkpointing interval

In the second thesis group I targeted to develop a new checkpointing algorithm based on workflow structure analysis that minimizes the checkpointing overhead while still keeping to the soft or hard deadline of the workflow. I have developed two algorithms, one that enables workflow level dynamism, and one that can realize a task level dynamism. My aim was also to show the connectivity between the sensitivity index of a workflow model and the effectiveness of the newly introduced algorithm.

3.3 Adaptive and user-steered execution

In the third thesis group we introduce iPoints, special control Points with the primary aim to help the scientist to interfere with the execution and according to provenance analysis to alter the workflow execution or to change the abstract model of the workflow. These iPoints are also capable to realize provenance based adaptive execution with the help of a so called Rule Based Engine (RBE) that can be controlled or updated by the scientist or with data mining support. In this thesis group I also give a specification of the above mentioned iPoints in IWIR (Interoperable Workflow Intermediate Representation) (Plankensteiner, Montagnat, and Prodan 2011) language, which was developed with the aim to enable interoperability between four existing SWfMSs within the framework of the SHIWA project. With this specification my aim was to support the user with the design and insertion of these special control points already in the composition phase.

4 Materials and Methods of Investigation

4.1 Fault sensitivity analysis and workflow structured based checkpointing algorithm

As a starting point of my research I thoroughly investigated the related work in the theme of faults and failures of dynamic execution. According to the reviewed literature I gave a taxonomy about most frequently arising failures during the workflow lifecycle (Bahsi 2008) (Gil et al. 2007) and about the existing solutions that were aimed to provide dynamic execution at a certain level.

The present dissertation employs two main methodologies to validate and evaluate the introduced formulas, ideas and algorithms. The first is an analytical approach. Taking into account that scientific workflows at the highest abstraction level are generally represented with Directed Acyclic Graphs, our validation technique is based on investigating the structure of the interconnected tasks.

As graphs can range in size from a few tasks to thousands of tasks, and values assigned to the edges and tasks may diverse, I started with simplifying the workflows with a transformation that eliminates the values assigned to the edges and homogenize the tasks. As a next step I used simple graph models to demonstrate my hypothesis, and afterward with use of algorithms and methods from the field of graph theory I demonstrated, validated and proved my results.

The second approach was to validate my results with simulations in Matlab, in a numerical computing environment by MathWorks. I have implemented algorithms for the invented formulas and for the checkpointing algorithms as well, and conducted numerous simulations based on special workflow patterns as well as on randomized workflows. For the randomized workflow patterns I took into account a survey on real-life workflows from the myExperiment.org website.

4.2 Provenance Based Runtime Manipulation

Studying the different scientific workflow management systems capabilities from the user steering perspective and collecting the emerging requirements for interfering we defined the set of tasks that should be supported when using intervention. After the detailed specification of the so called iPoint (intervention points), we implemented it in IWIR language.

5 New Scientific Results

Thesis group 1: Workflow structure analysis.

Thesis 1.1:

Thesis 1.1

I have defined the influenced zone of a task in a workflow represented with DAG, concerning to a certain time delay. Based on the influenced zones of the tasks I have defined the workflow sensitivity index which can help in fine-tuning the actually used fault tolerant method.

Thesis 1.2:

Thesis 1.2

I have developed an algorithm to calculate the influenced zone of a task and sensitivity index for complex graphs consisting of a high number of tasks and data dependencies. The time requirement of this algorithm is a polynomial function of the number of tasks and edges.

Thesis 1.3:

Thesis 1.3

I gave a classification for the workflows based on their workflow structure analysis.

Relevant own publications pertaining to this thesis group: [K-7; K-9; K-6; K-5; K-3; K-1]

Thesis group 2: Workflow structure based checkpointing algorithm
Thesis 2.2:

Thesis 2.1
I have developed a workflow-level, periodic (Wsb) checkpointing algorithms for DAG based scientific workflows, which can be used with known, constant checkpointing costs and known failure rate. The algorithms decreases the checkpointing overhead compared to the checkpointing algorithm optimized for the execution time, without affecting the total wallclock time of the workflow.

Thesis 2.2:

Thesis 2.2
I have developed the adaptive version of the proposed Wsb algorithm, which may further decrease the checkpointing overhead in the case when the real execution and data transmission time encounter some difference compared to the estimated ones. In this case the algorithm may also decrease the execution time of the workflow compared to the static (Wsb) algorithm.

Relevant own publications pertaining to this thesis group: [K-5; K-7; K-9]

Thesis 3. Provenance Based Runtime Manipulation.

Thesis 3.1:

Thesis 3.1
I have defined special control points, (iPoints), with the help of which and based on provenance analysis real-time adaptive execution of scientific workflows can be realized.

Thesis 3.2:

Thesis 3.2
I have defined special control points, (iPoints), with the help of which real-time user-steered execution of scientific workflows can be realized.

Thesis 3.3:

Thesis 3.3
I have specified the control points introduced in *thesis 3.1 and 3.2* in an Interoperable Workflow Intermediate Representation (IWIR) language.

Relevant own publications pertaining to this thesis group: [K-2; K-4; K-7; K-8]

6 Discussion and Practical Applicability of the Results

6.1 Workflow structure analysis and fault tolerance

The workflow structure analysis gives a detailed insight about the structure of complex graphs with high number of vertices and edges from a fault tolerant perspective. This analysis can be carried out in polynomial time of the number of vertices and edges.

As it was demonstrated in the second thesis group these results can be used to adjust checkpointing interval, and of course it can also be used to customize other fault tolerant methods, for example to determine the optimal number of replicas to be started according to the flexibility parameter and the sensitivity parameter for the individual tasks. The relationship between the results of the first and second thesis groups show us the significance of the workflow structure analysis.

These results can also be used during enactment to notify the user about the expected completion time according to the failures

6.2 Provenance based adaptive and user steered execution

As a result of the increasing need for user controlled and moreover the provenance based adaptive execution we introduced special control points where the user has the possibility to interfere with the workflow execution. With the help of the Rule Based Engine we also enable adaptive execution. The overall results can be used to guide the design and implementation of these special control points into the gUSE/WS-PGRADE system. For other SWfMSs it can also provide a practical guidance to follow by adapting it to their special definition language. The selected language for the specification would further promote interoperability of the workflow management systems.

Bibliography

References

Agarwal, Saurabh et al. (2004). “Adaptive incremental checkpointing for massively parallel systems”. In: *Proceedings of the 18th annual international conference on Supercomputing*. ACM, pp. 277–286.

Ailamaki, Anastasia (2011). “Managing scientific data: lessons, challenges, and opportunities”. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, pp. 1045–1046.

Bahsi, Emir Mahmut (2008). “Dynamic Workflow Management For Large Scale Scientific Applications”. PhD thesis. Citeseer.

Deelman, Ewa and Yolanda Gil (2006). “Managing Large-Scale Scientific Workflows in Distributed Environments: Experiences and Challenges.” In: *e-Science*, p. 144.

Di, Sheng et al. (2013). “Optimization of cloud task processing with checkpoint-restart mechanism”. In: *2013 SC-International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE, pp. 1–12.

Dias, Jonas et al. (2011). “Supporting dynamic parameter sweep in adaptive and user-steered workflow”. In: *Proceedings of the 6th workshop on Workflows in support of large-scale science*. ACM, pp. 31–36.

Fahringer, Thomas, Jun Qin, and Stefan Hainzer (2005). “Specification of grid workflow applications with AGWL: an Abstract Grid Workflow Language”. In: *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005*. Vol. 2. IEEE, pp. 676–685.

Garg, Ritu and A Kumar Singh (2011). “Fault tolerance in grid computing: state of the art and open issues”. In: *International Journal of Computer Science & Engineering Survey (IJCSES)* 2.1, pp. 88–97.

Gil, Yolanda et al. (2007). “Examining the challenges of scientific workflows”. In: *Ieee computer* 40.12, pp. 26–34.

Jhawar, Ravi, Vincenzo Piuri, and Marco Santambrogio (2013). “Fault tolerance management in cloud computing: A system-level perspective”. In: *IEEE Systems Journal* 7.2, pp. 288–297.

Kacsuk, Peter (2011). “P-GRADE portal family for grid infrastructures”. In: *Concurrency and Computation: Practice and Experience* 23.3, pp. 235–245.

Krinke, Jens (2001). “Identifying similar code with program dependence graphs”. In: *Reverse Engineering, 2001. Proceedings. Eighth Working Conference on*. IEEE, pp. 301–309.

Kwak, Seong Woo and Jung-Min Yang (2012). “Optimal checkpoint placement on real-time tasks with harmonic periods”. In: *Journal of Computer Science and Technology* 27.1, pp. 105–112.

Lee, Kevin, Norman W Paton, et al. (2009). “Adaptive workflow processing and execution in pegasus”. In: *Concurrency and Computation: Practice and Experience* 21.16, pp. 1965–1981.

Lee, Kevin, Rizos Sakellariou, et al. (2007). “Workflow adaptation as an autonomic computing problem”. In: *Proceedings of the 2nd workshop on Workflows in support of large-scale science.* ACM, pp. 29–34.

Li, Zhongwen and Hong Chen. “Adaptive Checkpointing Schemes for Fault Tolerance in Real-Time Systems with Task Duplication”. In:

Lidya, A et al. (2010). “Dynamic adaptation of checkpoints and rescheduling in grid computing”. In: *International Journal of Computer Applications (0975–8887)* 2.3.

Ludäscher, Bertram et al. (2009). “Scientific process automation and workflow management”. In: *Scientific Data Management: Challenges, Existing Technology, and Deployment, Computational Science Series* 230, pp. 476–508.

Meroufel, Bakhta and Ghalem Belalem (2014). “Adaptive time-based coordinated checkpointing for cloud computing workfl ows”. In: *Scalable Computing: Practice and Experience* 15.2, pp. 153–168.

Missier, Paolo et al. (2010). “Taverna, reloaded”. In: *International conference on scientific and statistical database management.* Springer, pp. 471–481.

Montagnat, Johan et al. (2009). “A data-driven workflow language for grids based on array programming principles”. In: *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science.* ACM, p. 7.

Nakagawa, Sayori, Satoshi Fukumoto, and Naohiro Ishii (2003). “Optimal checkpointing intervals of three error detection schemes by a double modular redundancy”. In: *Mathematical and computer modelling* 38.11, pp. 1357–1363.

Oliner, Adam J et al. (2005). “Performance implications of periodic checkpointing on large-scale cluster systems”. In: *19th IEEE International Parallel and Distributed Processing Symposium.* IEEE, 8–pp.

Oliveira, Daniel de et al. (2014). “Debugging Scientific Workflows with Provenance: Achievements and Lessons Learned”. In: *29th SBBD–SBBD Proceedings, Curitiba, PR, Brazil.*

Palaniswamy, Avinash C and Philip A Wilsey (1993). “An analytical comparison of periodic checkpointing and incremental state saving”. In: *ACM SIGSIM Simulation Digest.* Vol. 23. 1. ACM, pp. 127–134.

Pietri, Ilia et al. (2014). "A performance model to estimate execution time of scientific workflows on the cloud". In: *Workflows in Support of Large-Scale Science (WORKS), 2014 9th Workshop on.* IEEE, pp. 11–19.

Pintas, Julliano Trindade et al. (2013). "SciLightning: a cloud provenance-based event notification for parallel workflows". In: *International Conference on Service-Oriented Computing.* Springer, pp. 352–365.

Plankensteiner, Kassian, Johan Montagnat, and Radu Prodan (2011). "IWIR: a language enabling portability across grid workflow systems". In: *Proceedings of the 6th workshop on Workflows in support of large-scale science.* ACM, pp. 97–106.

Plankensteiner, Kassian, Radu Prodan, et al. (2013). "Fine-grain interoperability of scientific workflows in distributed computing infrastructures". In: *Journal of grid computing* 11.3, pp. 429–455.

Poola, Deepak et al. (2014). "Robust scheduling of scientific workflows with deadline and budget constraints in clouds". In: *2014 IEEE 28th International Conference on Advanced Information Networking and Applications.* IEEE, pp. 858–865.

Sakellariou, Rizos and Henan Zhao (2004). "A low-cost rescheduling policy for efficient mapping of workflows on grid systems". In: *Scientific Programming* 12.4, pp. 253–262.

Shi, Zhiao, Emmanuel Jeannot, and Jack J Dongarra (2006). "Robust task scheduling in non-deterministic heterogeneous computing systems". In: *2006 IEEE International Conference on Cluster Computing.* IEEE, pp. 1–10.

Starlinger, Johannes, Bryan Brancotte, et al. (2014). "Similarity search for scientific workflows". In: *Proceedings of the VLDB Endowment* 7.12, pp. 1143–1154.

Starlinger, Johannes, Sarah Cohen-Boulakia, et al. (2014). "Layer decomposition: An effective structure-based approach for scientific workflow similarity". In: *e-Science (e-Science), 2014 IEEE 10th International Conference on.* Vol. 1. IEEE, pp. 169–176.

Taylor, Ian et al. (2003). "Triana applications within grid computing and peer to peer environments". In: *Journal of Grid Computing* 1.2, pp. 199–217.

Team, SHIWA et al. (2011). *SHIWA: SHaring Interoperable Workflows for Large-Scale Scientific Simulation on Available DCIs.*

Terstyanszky, Gabor et al. (2014). "Enabling scientific workflow sharing through coarse-grained interoperability". In: *Future Generation Computer Systems* 37, pp. 46–59.

Turi, Daniele et al. (2007). "Taverna workflows: Syntax and semantics". In: *e-Science and Grid Computing, IEEE International Conference on.* IEEE, pp. 441–448.

Vahi, Karan et al. (2012). "A general approach to real-time workflow monitoring". In: *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:* IEEE, pp. 108–118.

Xiang, Xiaorong and Gregory Madey (2007). “Improving the Reuse of ScientificWorkflows and Their By-products”. In: *IEEE International Conference on Web Services (ICWS 2007)*. IEEE, pp. 792–799.

Young, John W (1974). “A first order approximation to the optimum checkpoint interval”. In: *Communications of the ACM* 17.9, pp. 530–531.

Own Publications Pertaining to Theses

K-1 Bánáti, Anna et al. (2015). “Usability of Scientific Workflow in Dynamically Changing Environment”. In: *Technological Innovation for Cloud-Based Engineering Systems: 6th IFIP WG 5.5/SOCOLNET Doctoral Conference on Computing, Electrical and Industrial Systems, DoCEIS 2015, Costa de Caparica, Portugal, April 13-15, 2015, Proceedings*. Ed. by M. Luis Camarinha-Matos et al. Springer International Publishing, pp. 129–136. ISBN: 9783319167664. DOI: 10.1007/978-3-319-16766-4_14. URL: http://dx.doi.org/10.1007/978-3-319-16766-4_14.

K-2 Kail, E, A Bánáti, et al. “Provenance based adaptive and dynamic workflows”. In: *15th IEEE International Symposium on Computational Intelligence and Informatics*, pp. 215–219.

K-3 Kail, Eszter, Anna Bánáti, et al. (2014). “Dynamic workflow support in gUSE”. In: *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*. IEEE, pp. 354–359.

K-4 Kail, Eszter, Péter Kacsuk, and Miklós Kozlovszky (2015a). “A novel approach to user-steering in scientific workflows”. In: *Applied Computational Intelligence and Informatics (SACI), 2015 IEEE 10th Jubilee International Symposium on*. IEEE, pp. 233–236.

K-5 — (2015b). “Achieving dynamic workflow management system by applying provenance based checkpointing method”. In: *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*. IEEE, pp. 250–253.

K-6 — (2015c). “New aspect of investigating fault sensitivity of scientific workflows”. In: *Intelligent Engineering Systems (INES), 2015 IEEE 19th International Conference on*. IEEE, pp. 185–188.

K-7 — (2016a). “A Novel Adaptive Checkpointing Method Based on Information Obtained from Workflow Structure”. In: *Transaction on Automating Control and Computer Science* 3.to be appear.

K-8 Kail, Eszter, Péter Kacsuk, and Miklós Kozlovszky (2016b). “Specification of user and provenance based adaptive control points at workflow composition level”. In:

International Symposium on Intelligent Sytems and Informatics (SISY), 2015 IEEE 14th. IEEE.

K-9 Kail, Eszter, Krisztián Karóczkai, et al. (2016). “Provenance Based Checkpointing Method for Dynamic Health Care Smart System”. In: *Scalable Computing: Practice and Experience* 17.2, pp. 143–153.

Own Publications Not Pertaining to Theses

Kx-1 Kail, E, S Khoor, K Fugedi, et al. (2005). “Expert system for phonocardiographic monitoring of heart failure patients based onwavelet analysis”. In: *Computers in Cardiology, 2005.* IEEE, pp. 833–836.

Kx-2 Kail, E, S Khoor, B Kail, et al. (2004). “Internet digital phonocardiography in clinical settings and in population screening”. In: *Computers in Cardiology, 2004.* IEEE, pp. 501–504.

Kx-3 Kail, E, S Khoor, and J Nieberl (2005). “Ambulatory wireless Internet electrocardiography: new concepts & maths”. In: *2nd International Conference on Broadband Networks, 2005.* IEEE, pp. 1001–1006.

Kx-4 Kail, Eszter, Gábor Németh, and Zoltán Richárd Turányi (2001a). “The effect of the transmission range on the capacity of ideal ad hoc networks”. In: *rN* 2, p. 3.

Kx-5 — (2001b). “Throughput of ideally routed wireless ad hoc networks”. In: *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing.* ACM, pp. 271–274.

Kx-6 Khoor, S et al. (2001). “Telemedicine ECG-telemetry with Bluetooth technology”. In: *Computers in Cardiology 2001.* IEEE, pp. 585–588.

Kx-7 Khoór, S et al. (2003). “Internet-based, GPRS, long-term ECG monitoring and non-linear heart-rate analysis for cardiovascular telemedicine management”. In: *Computers in Cardiology, 2003.* IEEE, pp. 209–212.

Kx-8 Rónai, Miklós Aurél and Eszter Kail (2003). “A simple neighbour discovery procedure for Bluetooth ad hoc networks”. In: *Global Telecommunications Conference, 2003. GLOBECOM’03. IEEE.* Vol. 2. IEEE, pp. 1028–1032.

Kx-9 Turányi, Zoltán R et al. (2000). “Global internet roaming with ROAMIP”. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 4.3, pp. 58–68.