HAL
open science

Robust Workflows for Large-Scale Multiphysics Simulation

Toan Nguyen, Laurentiu Trifan, Jean-Antoine Desideri

To cite this version:

Toan Nguyen, Laurentiu Trifan, Jean-Antoine Desideri. Robust Workflows for Large-Scale Multiphysics Simulation. Fifth European Conference on Computational Fluid Dynamics, Jun 2010, Lisbonne, Portugal. inria-00524660

HAL Id: inria-00524660

https://inria.hal.science/inria-00524660

Submitted on 8 Oct 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal, 14–17 June 2010

ROBUST WORKFLOWS FOR LARGE-SCALE MULTIPHYSICS SIMULATION

Toàn Nguyên, Laurentiu Trifan, Jean-Antoine Désidéri

INRIA
Centre de Grenoble Rhône-Alpes, 655, av. de l'Europe, Montbonnot,
38334 Saint-Ismier (France)
{Toan.Nguyen, trifan}@inrialpes.fr
Jean-Antoine.Desideri@sophia.inria.fr

Key words: Simulation, Workflows, Robustness, Multi-physics applications, Fault-tolerance

Abstract. *Large-scale simulations, e.g. fluid-structure interactions and aero-acoustics noise generation, require important computing power, visualization systems and high-end storage capacity. Because 3D multi-physics simulations also run long processes on large datasets, an important issue is the robustness of the computing systems involved, i.e., the ability to resume the inadvertantly aborted computations.*
A new approach is presented here to handle application failures. It is based on extensions of bracketing checkpoints usually implemented in database and transactional systems. An assymetric scheme is devised to reduce the number of checkpoints required to safely restart aborted applications when unexpected failures occur.
The tasks are controled by a workflow graph than can be deployed on various distributed platforms and high-performance infrastructures. An automated bracketing process inserts in the workflow graph checkpoints that are placed at critical execution points in the graph. The checkpoints are inserted using a heuristic process based on a evolving set of rules. Preliminary tests show that the number of checkpoints, hence the overhead incurred by the checkpointing mechanism, can be significantly reduced to enhance the application performance while supporting its resilience.

1 INTRODUCTION

The occurrence of random hardware and software failures is a challenging issue that raises many opportunities for smart solution in multi-core multi-processor systems. Critical application code is also the source of failures that can severely impact the processing of petascale volumes of data.

A new approach is presented here to handle random application failures. It is based on extensions of bracketing checkpoints usually implemented in database and transactional systems. An asymetric scheme is devised to reduce the number of checkpoints required to safely restart aborted applications when unexpected failures occur.

The approach is implemented as a plug-in for a workflow management system . This platform is compatible with remote parallel and high-performance computing infrastructures based on multi-core systems and supercomputers.

In this approach, multi-physics applications are modeled by interconnected graphs of distributed tasks. These tasks represent application codes. They can be distributed and run several instances of the code on remote locations where heterogeneous computing systems are running, including Beowulf clusters, supercomputers, multi-core multi-processors clusters, based on graphic processors and standard scalar processors.

Data replication and redundant code instances are some possible ways to consider fault-tolerance in high-performance computing systems [30]. Probabilistic approaches also support the provisioning for hardware and software failures. Incremental checkpointing facilities have recently been proposed to reduce the overhead incurred by checkpoint-restart mechanisms. Resilient workflows are considered here as a means to support large-scale dynamic and resilient multiphysics simulation and optimization applications, such as multiphysics aircraft simulation.

The paper is organized as follows. Section 2 details some characteristics of fault-tolerant and resilient workflows. It gives insights on the bracketing and rule mechanisms used to implement workflow resiliency. Section 3 details a prototype distributed platform that is implemented for a larger multidiscipline optimization project involving several academic and industry partners. Section 4 is a conclusion.

2 RESILIENT WORKFLOWS

Workflow management systems have recently been the focus of much interest and many research and deployment for scientific applications worldwide [18, 19, 27]. Their ability to abstract the applications by wrapping application codes have also stressed the usefulness of such systems for multidiscipline applications [15, 29]. When complex applications need to provide seamless interfaces hiding the technicalities of the computing infrastructures, their high-level modeling, monitoring and execution functionalities help giving production teams seamless and effective facilities [7, 17, 25]. Software integration infrastructures based on programming paradigms such as Python, Mathlab and Scilab have also provided evidence of the usefulness of such approaches for the tight coupling of multidisciplne application codes [14, 16]. Also high-performance computing based on multi-core multi-cluster infrastructures open new opportunities for more accurate, more extensive and effective robust multi-discipline simulations for the decades to come [20]. This supports the goal of full flight dynamics simulation for 3D aircraft models within the next decade, opening the way to virtual flight-tests and certification of aircraft in the future [15, 21].

2.1 Fault-tolerance

Because distributed systems are potentially faced with unexpected hardware and software failures, adequate mechanisms have been devised to handle recovery of running systems, software and applications.

Checkpoint and restart mechanisms are usually implemented using the local ordering of the running processes. This implies that the safe execution of all the running processes is not guaranteed, i.e., there is no way a randomly aborted distributed process can be restored in a consistent state and resume correctly. The solution would be to use a global synchronization and clock, which is practically unfeasible and very constraining.

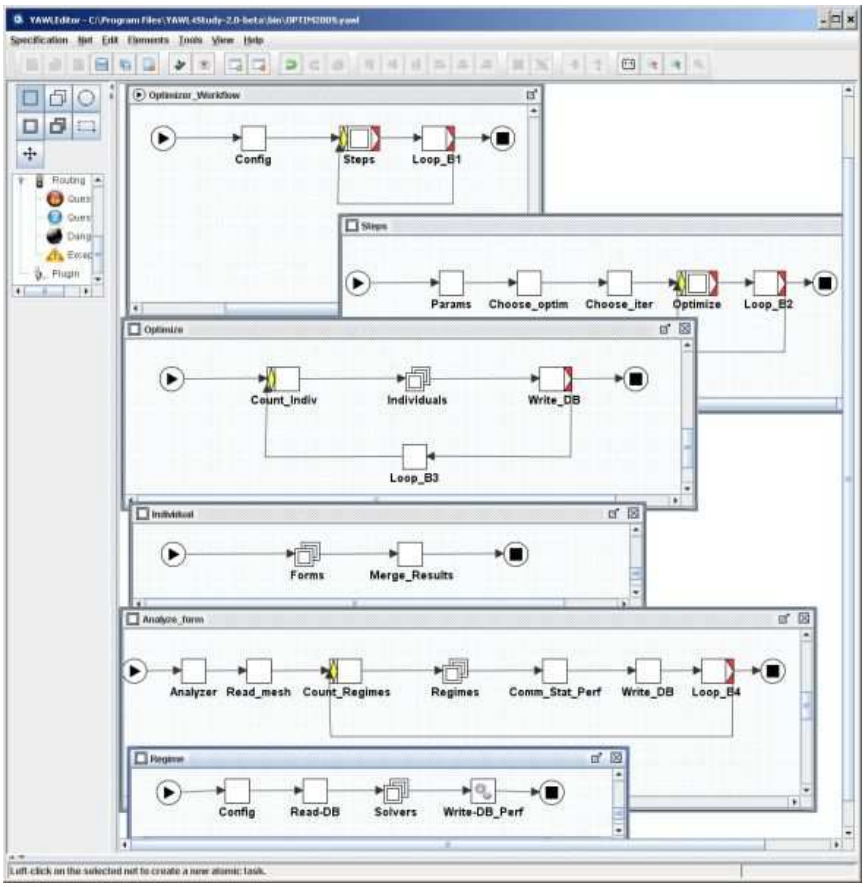

Figure 1. Composite workflow for wing aerodynamics optimization.

On the application side, this is used by transaction systems, e.g., airline reservation and banking systems, because compensation in case of failure is fundamental.

However, design, simulation and optimization applications bear specificities that require less stringent mechanisms than transaction systems. Design is a stepwise process that does not require global synchronizing, except when, and only when, dynamic update propagation is required. This can be executed during limited time periods and does not impair the usual stepwise approach.

2.2 Resilience

Resilience differs here from fault-tolerance. It is defined here as the ability of the applications to survive to unpredictable behavior. In contrast, fault-tolerance is defined as the ability to survive to hardware and system or communications failures.

In contrast with fault-tolerant workflows which can survive hardware and system failures, using ad-hoc bracketing by checkpointing mechanisms (Section 2.3), resilient workflows need to be aware of the application structure in order to implement automated survival procedures. These procedures can use the bracketing of sub-workflows also, but in addition, they need specific logging of the workflow component operations and parameters to restore incrementally previous states and resume partially their operations (Figure 3).

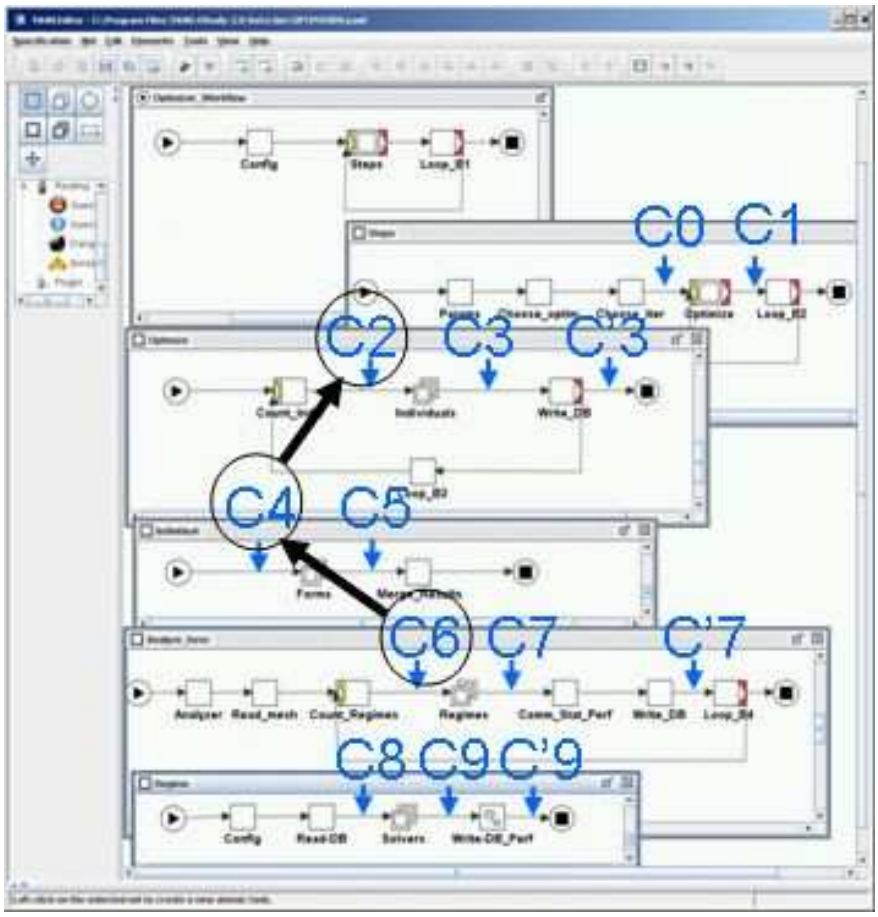

Figure 3. Resilient workflow: iterative recovery

2.3 Checkpoints

Thus, checkpoints must be inserted in the workflow composite hierarchy. They can bracket critical parts of the hierarchy, e.g., the most demanding CPU components (unsteady flow calculations over a 3D wing model, for example) and the following optimization components which might be less CPU demanding, but are fundamental to the application because they allow for the comparison of various optimized solutions. This scheme is called *bracketing checkpoints*.

Further, parallel branches of the workflow that failed need later to be re-synchronized with the branches that resumed correctly. This requires that the results of the successful branches are stored for further processing with the failed branches results, if they resume correctly later. Otherwise, these results are discarded if the failed branches never succeed. Because there is no awareness of the successful branches on the possible failures of parallel branches in the workflow, time-out and synchronization signals must

be exchanged on a regular basis to notify each branch of the current state of the others: alive or not responding.

2.4 Rules

We assume in the following that "join" operations are those that require several input datastreams to execute. Similarly, we assume that "fork" operations are those that output their results on several datastreams. They model generic tasks that execute application codes. We also consider "remote" and "local" operations. We do not distinguish between parallel and sequential implementations of the operations.

Further, we consider in the following that the "specified" operations are those operations or workflow tasks that are marked by the application designers or the users as requiring a specific treatment in the following heuristic procedure.

The specific characterization of the marked tasks is implemented by raising an exception that invokes a specific treatment that departs from the standard heuristic rules. An example of such exception is the backup of a particular intermediate result after processing by a large CPU intensive task or the back up of the result of a task producing petascale volumes of data. Workflow management systems usually provide powerful exception handling functionalities that can be used to implement this kind of "specified" operations management, e.g., YAWL [11].

This enables the designers and users to adapt the execution of the workflow depending on their specific knowledge and expertise.

This is a prerequisite for the effective implementation of the workflows based on previous runs and casestudies involving petaflops and petabytes of data. Some automated learning procedure could eventually be designed to support this kind of feedback..

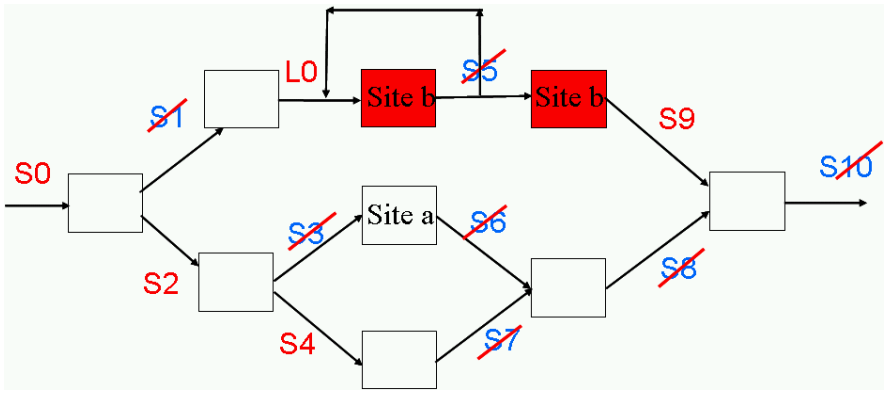

Figure 4. Asymmetric checkpoints.

The recovery procedure implements a heuristic approach based on the following rules:

- R1: no output backup for specified join operations
- R2: only one output backup for fork operations
- R3: no intermediate result backup for user-specified sequences of operations
- R4: no backup for user-specified local operations
- R5: systematic backup for remote inputs

In order to improve performance, these rules can be tuned by the application designers to fit their specific characteristics and requirements. This includes specified operations that are deemed CPU intensive and data intensive. Also, distributed scheduling policies can be taken into account through such rules.

However, the current prototype implementation described in Section 3 relies on a middleware that provides for resource allocation and load-balancing. Therefore, the rules can be altered or ignored by load-balancing strategies if appropriately authorized by the designers and users, and also if global and local policies make this mandatory, e.g., preemptive local strategies.

Based on these rules, the example illustrates the asymmetric cascading checkpoints on an unfolded workflow (Figure 4). Two remote execution sites are considered: Site a (white colored tasks) and Site b (red colored tasks).

When it is not modified and tuned by the designers, the result of the asymmetric cascading checkpoints procedure results in seven unnecessary checkpoints which are deleted, thus leaving five remaining checkpoints: S0, S2, S4, S9 and L0.

3 IMPLEMENTATION

3.1 Distributed workflow platform

The approach implemented here uses the YAWL workflow management system. It is one of the rare workflow systems to be defined with a sound formal semantics [4, 13]. The platform is designed to combine grid and distributed computing through a middleware with scientific computing using a mathematical problem solving environment. It thus provides an e-Science infrastructure as a high-performance platform for large-scale distributed data and CPU intensive applications. Validation of the platform is through industrial testcases concerning car aerodynamics and engine valves and pipes optimization.

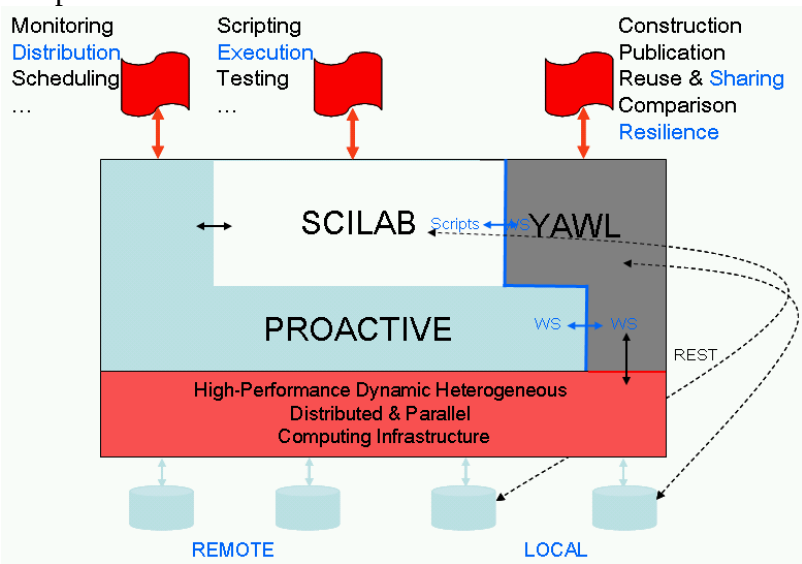

Figure 5. OMD2 distributed optimization platform.

The approach wraps the existing applications codes, e.g., optimizers and solvers, with Web services that are invoked through Web services invocations.

The rules defined in Section 2.4 are implemented in a prototype distributed platform using the YAWL workflow management system for application composition, monitoring and scheduling.

The resiliency procedure uses default and user-defined rules as explained above. They are implemented using the worklet service provided by the YAWL system. It is called the “Worklet dynamic process selection and exception handling service for YAWL” [32]. It is a powerful exception handling and rule mechanism which uses a

dynamic selection and invocation mechanism including hierarchically defined rules [31].

Worklets are exception handlers. They are defined by standard workflows. Worklets are substituted dynamically at run-time for the YAWL work item, i.e., the application high-level tasks, that raised the exceptions (time-outs, abnormal behaviors, checkpointing, etc). They are invoked using the original item parameters and execute specific workflows that are stored in a dynamically updated directory of Web services. Also, several worklets can be launched simultaneously following an exception raising. Finally, new rules can be added dynamically and inserted in the exception handling mechanism. Thus an executing worklet can be dynamically replaced by another one. The system supports also recursive worklets and external events triggered by the users, providing a very powerful mechanism for dynamically changing the applications behavior at run-time [32].

Rules are specific elements handled by an appropriate mechanism. They allow for the application to pause, abort, resume and restart, based on the exceptions raised. Combined with the ability to add new exceptions on-line, and the handling worklets, this provides for a context (data) dependent behavior of the applications.

This allows for example different optimization algorithms to be dynamically invoked depending on the data being processed and on the convergence of the optimizers. It also allows for different exception handling treatments to be invoked, e.g., abort, restart, backtrack, suspend, recover, forced completion, forced failure, etc.

Further, as new exceptions occur, rules can be dynamically added by the users to the repository without modification of the original workflow specification. They become part of the workflow definition for future executions. This is a seamless approach to adapt the application behavior to unforeseen situations.

A platform supporting these features is developed for the OMD2 project [21] by a consortium that includes twelve academic and industry partners, including a major international car manufacturer leading the project. OMD2 is a French acronym for Distributed Multi-Discipline Optimization.

The goal is to develop a high-performance distributed environment for simulation and multidiscipline optimization for complex design projects.

3.2 Multidiscipline platform

The distributed platform uses the ProActive middleware for resource allocation and scheduling of tasks [19]. The tasks include software codes that collaborate and include Scilab scripts [20], optimization software developed by the project partners as well as commercial CAD tools.

YAWL is used for defining incrementrally composite workflows, as well as the sharing and reuse of the various software that form the applications (Figure 5). These can interact with the users through sophisticated exception handling mechanisms [31] and interact with each other using Web services [28]. This is also used for implementing the resilience and fault-tolerance features described in Section 2.

Because the workflow engine supports dynamic interactions with external software through Web Services, it communicates with the ProActive engine using specific services for distributed resource allocation and scheduling of the applications. Similarly, interaction with the Scilab numeric computation software is based on script invocations for the execution of the application codes. Access to local data and codes are available for both Scilab and YAWL users. Specific CAD, CFD, design and optimization tools used by the partners of the OMD2 project are also connected to the platform, e.g., CATIA, OpenFOAM, StarCD, ModeFrontier, as well as a SciLab optimization toolkit

that includes the partners' software. They benefit therefore from the features provided by the three platform components for application, data sharing and reuse as well as distributed execution:
- the Scilab scientific software package for numerical computations,
- the ProActive middleware for distributed task allocation, execution and monitoring,
- the YAWL workflow system for application definition, evolution and sharing.

4 CONCLUSION

Distributed infrastructures exhibit potential hazards to the executing processes, due to unexpected hardware and software failures. This is endangered by the use of distributed high-performance environments that include very large clusters of multi-processors nodes. This requires fault-tolerant workflows systems.

Further, erratic application behavior requires dynamic user interventions, in order to adapt execution parameters for the executing codes and to run dynamic application re-configurations. This requires resilient workflow systems.

This implies applications roll-back to appropriate checkpoints, and the implementation of survivability procedures, including fault-tolerance to external failures and resiliency to unexpected application behavior.

Asymmetric cascading checkpoints are presented here to effectively support the resiliency procedure. In order to minimize the overhead incurred by the checkpointing and logging of the workflow operations, a heuristic is presented that uses tunable rules to adapt the resiliency procedure to the application requirements and particular context at run-time.

Several open issues are currently under investigation, including the impact of the rule ordering on the resilience performance in case of application restart after unexpected crashes, the impact of user-defined rules inserted in the default rule set and the impact of application characteristics (CPU and data intensive) on the expected resilience/application performance ratio.

ACKNOWLEDGMENTS

This work is the result of the authors' contributions to:

- the OPALE project at INRIA: http://www-opale.inrialpes.fr,
- the AEROCHINA2 project of the EC, supported by the "Transport including Aeronautics" program of the FP7, contract n°ACS7-GA-2008-213599 (www.cimne.com/aerochina2),
- the OMD2 project ("Optimisation Multi-Disciplines Distribuée") of the French National Research Agency (ANR), grant n°ANR-08-COSI-007-07, program COSINUS ("Conception et Simulation").

The authors wish to thank all their partners for their support and Prof. Jacques Périaux from CIMNE (Barcelona, Spain) for fruitful discussions.

REFERENCES

[1] Y.Simmhan et al. "Building the Trident Scientific Workflow Workbench for Data Management in the Cloud". In proceedings of the *3rd Intl. Conf. on Advanced Engineering Computing and Applications in Science*. ADVCOMP'2009. Sliema (Malta). October 2009.

[2] A. Abbas, High Computing Power: A radical Change in Aircraft Design Process, In proceedings of the *2nd China-EU Workshop on Multi-Physics and RTD Collaboration in Aeronautics*. Harbin (China) April 2009.

[3] T. Nguyên and J-A Désidéri, Dynamic Resilient Workflows for Collaborative Design, In proceedings of the 6th *Intl. Conf. on Cooperative Design, Visualization and Engineering*. Luxemburg. September 2009. Springer-Verlag. *LNCS 5738*, pp. 341–350 (2009)

[4] W. Van der Aalst et al., Modern Business Process Automation: YAWL and its support environment, *Springer* (2010).

[5] B. Abou El Majd, J.-A.Désidéri, and R. Duvigneau, Multilevel strategies for parametric shape optimization in aerodynamics, *European Journal of Computational Mechanics* **17**, 1–2 (2008).

[6] D. Mogilevsky et al., Byzantine anomaly testing in Charm++: providing fault-tolerance and survivability for Charm++ empowered clusters, In proceedings of the *6th IEEE Intl. Symp. On Cluster Computing and Grid Workshops*. CCGRIDW'06. Singapore. May 2006.

[7] E. Deelman et Y. Gil., Managing Large-Scale Scientific Workflows in Distributed Environments: Experiences and Challenges, In proceedings of the *2nd IEEE Intl. Conf. on e-Science and the Grid*. Amsterdam (NL). December 2006.

[8] V. Selmin, Advanced Tools for Multidisciplinary Analysis and Optimisation, In proceedings of the *2nd China-EU Workshop on Multi-Physics and RTD Collaboration in Aeronautics*. Harbin (China) April 2009.

[9] T. Nguyên, A perspective on High-Performance Collaborative Platforms for Multidiscipline Simulation and Optimization, In proceedings of the *2nd China-EU Workshop on Multi-Physics and RTD Collaboration in Aeronautics*. Harbin (China). April 2009.

[10] Y. Simmhan et al., GrayWulf: Scalable Softawre Architecture for Data Intensive Computing, In proceedings of the *42nd Hawaii Intl. Conf. on System Sciences*. January 2009.

[11] W. van der Aalst et al., Design and implementation of the YAWL sytem, In proceedings of the *16th Intl. Conf. on Advanced Information Systems Engineering. CaiSE'2004*. 2004.

[12] M. Adams et al., YAWL User Manual, Version 2.0f, *The YAWL Foundation*. September 2009.

[13] N. Russel et al.., Workflow control flow patterns. A revised view, *Tech. report. University of Eindhoven (NL)*. 2006.

[14] A.M. Wissink et al., A Multi-code Python-based Infrastructure for Overset CFD with Adaptive Cartesian Grids, In proceedings of the *46th AIAA Aerospace Science Meeting* . Reno (Nevada). January 2008.

[15] D.J. Mavripilis et al., Petaflops Opportunities for the NASA Fundamental Aeronautics Program, In proceedings of the *18th AIAA Computational Fluid Dynamics Conference*. Miami (Florida). June 2007.

[16] J. Klenner et al., Future simulation Concept, In proceedings of the *1st European CEAS Air and Space Conference*. Berlin (G). September 2007.

[17] D.J. Mavripilis et al., High-Performance Computational Engineering: Putting the "E" Back in CSE, In proceedings of the *21st Intl. Conf. on Parallel Fluid Dynamics*. Moffett Filed (Ca.). May 2009.

[18] E. Deelman et al. Proceedings of the *NSF Workshop on the Challenges of Scientific Workflows*. Arlington (Va.). May 2006.

[19] Y. Gil et al., Examining the challenges of Scientific Workflows, *IEEE Computer*. December 2007.

[20] H. Simon., Future directions in High-Performance Computing 2009-2018, Invited lecture given at the *ParCFD 2009 Conference*. Moffett Field (Ca). May 2009.

[21] G. Katsaros et al., CFD Automatic Optimization using OpenFOAM in Grid Environments, In proceedings of the *OpenFOAM Intl. Conf.* 2007. London (U.K.). November 2007.

[22] M. Ghanem et al., Grid-enabled workflows for industrial product design, In proceedings of the *2nd Intl. Conf. on e-Science and Grid Computing*. Amsterdam (NL). December 2006.

[23] G. Kandaswamy et al., Fault-tolerant and recovery of scientific workflows on computational grids, In proceedings of the *8th Intl. Symp. On Cluster Computing and the Grid*. 2008.

[24] I.C. Kampolis et al., CFD-based analysis and two-level aerodynamic optimization on graphics processing units, *Paper submitted for publication*. September 2009.

[25] J. Wang et al., A high-level distributed execution framework for scientific workflows, In proceedings of the *4th IEEE Intl. Conf. on eScience*. Indianapolis (In). December 2008.

[26] G. Martinez et al., Integrating scheduling policies into workflow engines, *Paper submitted for publication*.

[27] J. Yu and R. Buyya., A taxonomy of workflow management systems for grid computing, In proceedings of the *SIGMOD 2005*. pp 44-49. New-York (USA).

[28] L. Aldred et al., Connecting custom services to the YAWL engine, Beta-7 Release, Technical Report, Faculty of Information Technology, Queensland University of Technology, Brisbane (Aus.), March 2006.

[29] A. Schreiber et al., Scientific Data and Knowledge Management in Aerospace Engineering, In proceedings of the *3rd Intl. Conf. on Advanced Engineering Computing and Applications in Sciences*. ADVCOMP2009. Sliema (Malta). October 2009.

[30] D. Crawl and I. Altintas, A Provenance-Based Fault Tolerance Mechanism for Scientific Workflows, In proceedings of the *2nd Intl. Provenance and Annotation Workshop*. IPAW 2008. Salt Lake City (UT). June 2008. Springer. LNCS 5272. pp 152-159.

[31] M. Adams et al., Facilitating Flexibility and Dynamic Exception Handling in Workflows throughWorklets, Technical report, Faculty of Information Technology, Queensland University of Technology, Brisbane (Aus.), October 2006.

[32] M. Adams et al., The worklet custom service for YAWL, Installation and User Manual, Beta-8 Release, Technical Report, Faculty of Information Technology, Queensland University of Technology, Brisbane (Aus.), October 2006.

[33] J. Qin et al., A Novel Graph Based Approach for Automatic Composition of High Quality Grid Workflows, In proceedings of the *18th Intl. Symposium on High Performance Distributed Computing*, Munich (G), June 2009.