ABSTRACT

Workflows have been successfully applied to express the decomposition of complex scientific applications. This has motivated many initiatives that have been developing scientific workflow tools. However, the existing tools still lack adequate support to important aspects namely, decoupling the enactment engine from workflow tasks specification, decentralizing the control of workflow activities, and allowing their tasks to run autonomous in distributed infrastructures, for instance on Clouds. Furthermore, many workflow tools only support the execution of Direct Acyclic Graphs (DAG) without the concept of iterations, where activities are executed millions of iterations during long periods of time and supporting dynamic workflow reconfigurations after certain iteration. We present the AWARD (Autonomic Workflow Activities Reconfigurable and Dynamic) model of computation, based on the Process Networks model, where the workflow activities (AWA) are autonomic processes with independent control that can run in parallel on distributed infrastructures, e.g., on Clouds. Each AWA executes a Task developed as a Java class that implements a generic interface allowing end-users to code their applications without concerns for low-level details. The data-driven coordination of AWA interactions is based on a shared tuple space that also enables support to dynamic workflow reconfiguration and monitoring of the execution of workflows. We describe how AWARD supports dynamic reconfiguration and discuss typical workflow reconfiguration scenarios. For evaluation we describe experimental results of AWARD workflow executions in several application scenarios, mapped to a small dedicated cluster and the Amazon (Elastic Computing EC2) Cloud.

Keywords: Autonomic Computing, Cloud, Parallel and Distributed Processing, Scientific Workflows, Tuple Space

AUTONOMIC WORKFLOW ACTIVITIES USING THE AWARD FRAMEWORK

The emergence of e-Science initiatives (Foster & Kesselman, 2003; Hine, 2006; NeSC, 2012) requires improved software tools and environments towards more productive modeling and experimentation with physical and virtual phenomena in diverse application domains. This involves complex computational processes for simulation, visualization, access to large data sets, and increasing degrees of interaction with the user.

DOI: 10.4018/ijaras.2014040104
The workflow paradigm, early adopted in the business context (Hollingsworth, 1995), is also useful for modeling scientific applications and allowing flexible mappings between the application abstractions and the underlying computing infrastructures. Workflows have been used for developing scientific applications in many domains (Taylor et al., 2007; Yu & Buyya, 2005; Deelman et al., 2005; Chin et al., 2011). Such efforts have been supported by multiple workflow tools (Lazweski, 2011), as Triana (2011), Taverna (2011) and Kepler (Kepler project, 2011). However, there is still a need for improving the support provided by the workflow tools, as discussed for example in Deelman (2007), Deelman et al. (2009), Chin et al. (2011) and Assuncao et al. (2009):

1. There is a need for a more clear separation of the application logic from the workflow enactment. The application developer often needs to understand details at the level of workflow engine. For example in Kepler, in order to develop a new Actor, the programmer needs to implement methods that are difficult to understand and are dependent on the enactment engine, e.g. `preinitialize(); initialize(); prefire(); fire(); postfire()` and `wrapUp()` (Kepler project, 2011);

2. Although multiple workflows models allow parallel execution, e.g. by following the Process Networks (PN) model (Kahn, 1974), (Parks, 1995), as in Kepler (PN Director) (Kepler project, 2011) and in (Agun & Bowers, 2011), however in such systems activities are usually executed as threads within the same process, which then acts as a monolithic workflow engine. Furthermore, a unique central entity as the Kepler Director handles the global coordination. In order to enable a more flexible workflow management and easing the mappings of the workflow activities across Clusters, Grids, and Clouds, there is a need for more decentralized control and coordination of the workflow activities, designed as autonomic components with separate control and behavior;

3. Flexibility in workflow management has been a long-term concern in business workflows (Dadam & Reichert, 2009), (Halliday et al., 2001), (Burkhart & Loos, 2010), but it has also been a trend towards enabling more complex scientific experiments. For example, allowing separate parts of a workflow to be launched or controlled individually by different users. Also allowing dynamic workflow reconfiguration, in response to changes in the application logic and behavior, e.g., in computational steering experiments, or interactive workflows. Although several approaches have been proposed (Ngu et al., 2008), (Yu & Buyya, 2010), there are still insufficient experimentation reports illustrating the feasibility of the above dimensions in real applications.

In previous work, we developed the AWARD (Autonomic Workflow Activities, Reconfigurable and Dynamic) model (Assuncao et al., 2012), to support the practical evaluation of solutions to the above concerns, in distinct application scenarios (Goncalves et al., 2012). AWARD has the following main characteristics: i) AWARD follows the Process Networks (PN) model, extended with autonomic processes for modeling the workflow activities (AWA), with decentralized control and allowing flexible mappings to distributed infrastructures such as the Cloud (Amazon EC2); ii) The AWARD shared space is used for data-driven communication and coordination of AWA activities as well as for dynamic reconfigurations; iii) End-users do not need to be aware of details on the enactment engine for developing the application algorithms (Tasks). These Tasks are developed as Java classes with a generic interface that can encapsulate calls to Web services, and run local or remote applications, supporting binary legacy applications in other languages, e.g., C or Fortran.
Related Content

Ways to View the World: A Standard Ontology as the Reality Framework and the World Code
www.igi-global.com/chapter/ways-view-world/28309?camid=4v1a

Timely Autonomic Adaptation of Publish/Subscribe Middleware in Dynamic Environments
www.igi-global.com/article/timely-autonomic-adaptation-publish-subscribe/62845?camid=4v1a

On the Privacy and Utility of Anonymized Social Networks
www.igi-global.com/article/privacy-utility-anonymized-social-networks/77630?camid=4v1a
Improving User Profiling for a Richer Personalization: Modeling Context in E-Learning