The Simple API for Grid Applications: Supporting e-Science by Providing Simple, Stable and Generic Programming Interfaces

Joao Abecasis[1], Shantenu Jha[1,2,3], Hartmut Kaiser[1],
Andre Merzky[1], Ole Weidner[1]

[1]*Center for Computation & Technology, Louisiana State University, USA*

[2]*Department of Computer Science, Louisiana State University, USA*

[3]*e-Science Institute, University of Edinburgh, UK*

May 15, 2008

SAGA is a high level API that provides a simple, standard and uniform interface to the most commonly required distributed functionality (see Figure 1). As shown in Figure 2, SAGA can be used for Grid applications [1, 2], tool-kits that manage distributed applications, as well as implement abstractions that support commonly occurring programming, access and usage patterns [3]. SAGA is currently an OGF approved technical specification [4], on the road map to becoming a standard. Although the process that has lead to the SAGA specification is community driven, the effort to validate, develop a specification-compliant implementation and to provide effective adaptors to make SAGA usable on different middleware and distributed systems has received a major boost by OMII-UK's decision to support the SAGA effort.

The aim of this paper is to provide a brief introduction to SAGA and show how it is an effective tool for developing e-Science applications. We will briefly discuss the sophisticated software development environments and tools that are used by the SAGA team and which are necessary, in order to provide simple, stable and usable implementations of SAGA. Finally, we will provide a summary and a roadmap of the many SAGA activities that are being undertaken by the SAGA team centered out of the Center for Computation and Technology (CCT) at Louisiana State University (LSU) as part of the OMIISAGA project[1]. In particular, we will show how, OMIISAGA in conjunction with other funded projects, for example, a project to develop Condor adaptors, which will bring SAGA to Campus Grids and many other high-throughput systems, is making an impact at every level of the software landscape required for distributed system – starting from the application level and downwards.

References

[1] S. Jha, H. Kaiser, Y. El Khamra, and O. Weidner, "Design and implementation of network performance aware applications using SAGA and Cactus," in *Accepted for 3rd IEEE Conference on eScience2007 and Grid Computing, Bangalore, India.*, 2007. [Online]. Available: http://saga.cct.lsu.edu/publications/saga_cactus_escience.pdf

[2] Developing Large-Scale Adaptive Scientific Applications with Hard to Predict Runtime Resource Requirements, *Proceedings of TeraGrid08*, available at http://tinyurl.com/5du32j.

[3] Exposing the Power of Google through SAGA, *Google Summer of Code* http://www.omii.ac.uk/wiki/MPGoogleSAGA.

[4] OGF Document Series 90, http://www.ogf.org/documents/GFD.90.pdf.

[1] http://saga.cct.lsu.edu

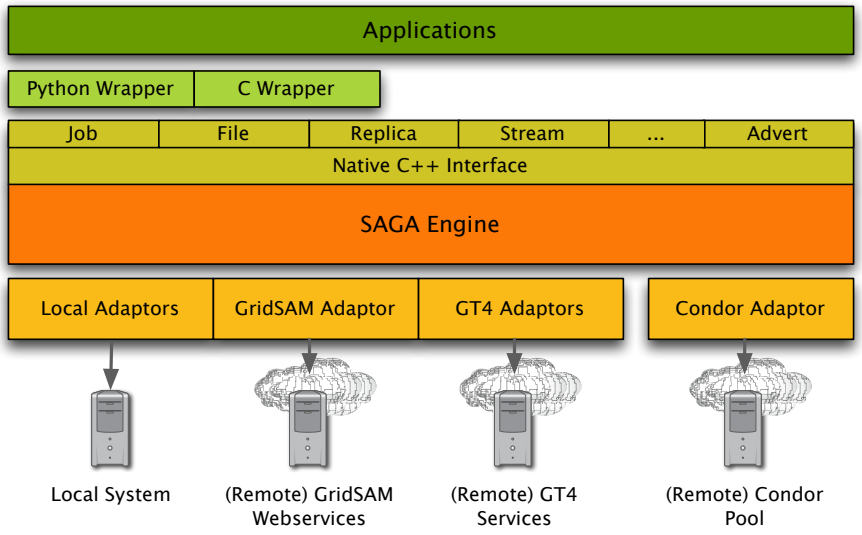

Figure 1: A layered diagram of how a typical application uses the C++ SAGA implementation (possibly via the C or Python wrappers) to implement the most commonly required functionality in distributed environments. The adaptors enable SAGA to be used to different middleware distributions; adaptors are typically loaded run-time and thus enable applications to be written once, but deployable on any system for which the SAGA adaptors exist.

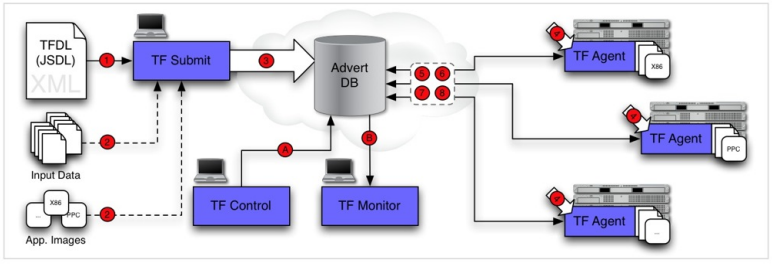

Figure 2: A schematic representation of how SAGA can be used to orchestrate the distributed resources and provide the required functionality. In this case SAGA is used to control the many stages of a task-farming application – a very simple, single stage workflow.