A 90 Minute *SAGA* Hands-On Tutorial

ISSGC, Nice, France, July 5-17 2009

May 24, 2009

Scope of this Tutorial

The scope of this tutorial is to provide the audience with the required resources and technical knowledge (hands-on experience) to start hacking their own distributed applications with SAGA.

Prerequisites

This tutorial requires basic knowledge of the C/C++ programming language. Experience using the command line on a Linux/UNIX based operating system and a basic idea of what a compiler, a linker and a Makefile is might come in handy.

Unless this tutorial is going to be preceded by a SAGA installation tutorial, the students are required to have a fully working installation of SAGA on their laptops/lab machines (preferred), or remote access (e.g. via SSH) to a machine with SAGA installed.

Cheat Sheat

At the beginning of the tutorial we're going to hand out the SAGA Cheat Sheet (yet to be created). The cheat sheet is a double-sided (laminated, so people won't throw it away!) letter-sized piece of paper with hints and tips for writing, compiling, and running SAGA applications.

Unit I (15 min)

A minimal SAGA application. Every application that uses just a single SAGA call is considered a SAGA applications, since it triggers the whole stack. SAGA is not a framework. It doesnt impose a specific programming model or way to use it (like, e.g. MPI). Look at it as a TOOLBOX for distributed computing.

`<CODE>`

Unit II (15 min)

Compiling and linking. Demonstrate how to include the SAGA Makefiles to compile and link a SAGA application. Explain the concept of packages as independent shared libraries. Demonstrate static vs. dynamic linking.

`<CODE>`

Unit III (15 min)

Running a SAGA application. Explain what needs to be in the loader path. Explain the effects of SAGA_VERBOSE and how it can be used to debug a saga application. Demonstrate how the engine launches adaptor loading, etc... in the background.

`<CODE>`

Unit IV (20 min)

Hello distributed world! Submit three jobs to three machines. One returns Hello, one returns Distributed and one returns World. They may or may not return in the right order. This should give the student an idea how they could potentially speed up their application using multiple resources.

`<CODE>`

Excercise (20 min)

Show what you've learned! Have the students write a distributed application We have to come up with something neat, fun!!!, and simple enough that can be solved by all students within 20 minutes and the help of the cheat sheet.

```
<CODE>
```

Conclusion