Apache Airavata Application Factory

Lahiru Gunathilake, Suresh Marru, Marlon Pierce
Indiana University
{lginnali,smarru,mpierce}@iu.edu

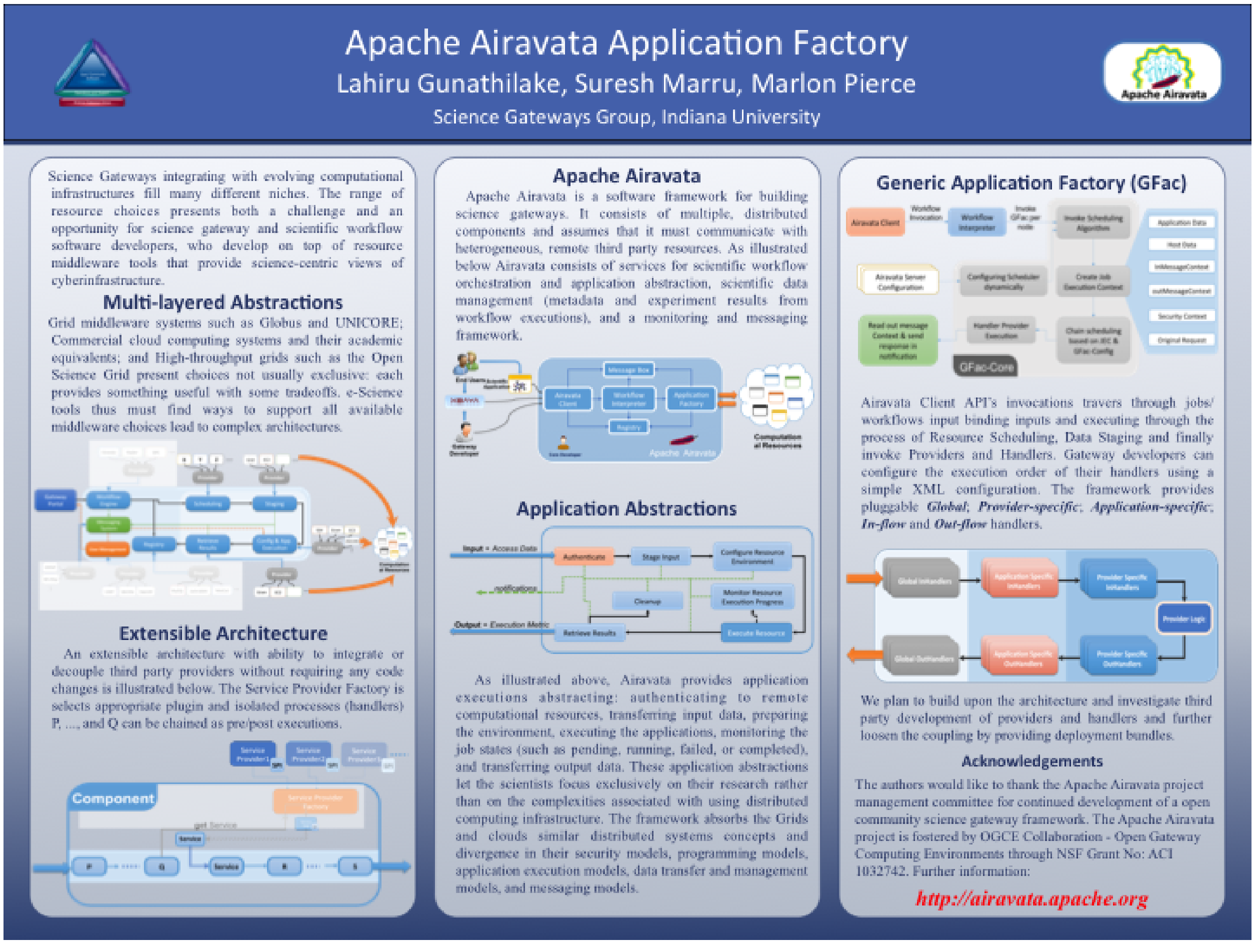
Apache Airavata's is an open source science gateway framework providing security, data, metadata, and job and workflow management capabilities for client gateways. The Generic Application Factory (GFac) is a component within Airavata that abstracts the details of job submission, monitoring, data transfer and error handling when interacting with remote computing resources running a variety of middleware. In this poster, we present a flexible and extensible architecture for GFac that allows it to work as a framework for managing clients to a wide range of third party middleware providers.

GFac's central design concept is the use of "providers" and "handlers", which are analogous to message body and message header processors in message-oriented systems. GFac's framework functionality is thus to handle multiple providers, each modified by zero or more handlers. The framework provides thread management, fault tolerance, and extensibility, while the handlers and providers implement the specific features needed to execute a task on a specific resource. As an example, a Grid application execution consists of atomic operations like authentication, input file transfer, job submission, job monitoring and output file transfer. All these steps can be implemented as multiple handlers to the application execution framework and the order can be configured in a configuration file. These handlers can be configured to run in synchronous or asynchronous mode. If handlers are asynchronous, they will be run in a separate thread which will be picked from the framework thread pool.

In our design, we must carefully consider the responsibility separation between framework and extensions. As an example, a use case like batching job submissions to a given host, bulk monitoring is orchestrated by framework but enacted by handlers. Apache Airavata has nearly 30 handlers implemented for different use cases. Users can organize the already existing handlers and reuse them in their applications or they can write their own handlers and organize them accordingly. At this time of this writing, GFac bundles support for GSISSH authentication, GSISCP file transfers, SSH Authentication, SCP File transfer , Job submission with torque based implementations like (qsub, squeue) and Job submission with SGE(Sun Grid Engine) and monitoring in SGE.

As a framework, GFac strives to provide inherent characteristics of **Extensibility**, **Fault Tolerance, Scalability (**across large number of application executions and related operations). **Flexibility** (avoiding deadlocks/starving or conflicting states), **Persistence** and **Dynamism**. The focus of the poster will be to discuss these intrinsic capabilities and lessons learnt while implementing these challenges.

The following figure illustrates a draft version of the poster which will be modified and updated by the time of the conference.



Apache Airavata Application Factory

Lahiru Gunathilake, Suresh Marru, Marlon Pierce

Science Gateways Group, Indiana University

Science Gateways integrating with evolving computational infrastructures fill many different niches. The range of resource choices presents both a challenge and an opportunity for science gateway and scientific workflow software developers, who develop on top of resource middleware tools that provide science-centric views of cyberinfrastructure.

Multi-layered Abstractions

Grid middleware systems such as Globus and UNICORE; Commercial cloud computing systems and their academic equivalents; and High-throughput grids such as the Open Science Grid present choices not usually exclusive: each provides something useful with some tradeoffs. e-Science tools thus must find ways to support all available middleware choices lead to complex architectures.

Extensible Architecture

An extensible architecture with ability to integrate or decouple third party providers without requiring any code changes is illustrated below. The Service Provider Factory is selects appropriate plugin and isolated processes (handlers) P, ..., and Q can be chained as pre/post executions.

Apache Airavata

Apache Airavata is a software framework for building science gateways. It consists of multiple, distributed components and assumes that it must communicate with heterogeneous, remote third party resources. As illustrated below Airavata consists of services for scientific workflow orchestration and application abstraction, scientific data management (metadata and experiment results from workflow executions), and a monitoring and messaging framework.

Application Abstractions

As illustrated above, Airavata provides application executions abstracting: authenticating to remote computational resources, transferring input data, preparing the environment, executing the applications, monitoring the job states (such as pending, running, failed, or completed), and transferring output data. These application abstractions let the scientists focus exclusively on their research rather than on the complexities associated with using distributed computing infrastructure. The framework absorbs the Grids and clouds similar distributed systems concepts and divergence in their security models, programming models, application execution models, data transfer and management models, and messaging models.

Generic Application Factory (GFac)

Airavata Client
Workflow Invocation
Workflow Interpreter
Invoke GFac per node
Invoke Scheduling Algorithm
Airavata Server Configuration
Configuring Scheduler dynamically
Create Job Execution Context
Read out message Context & send response in notification
Handler Provider Execution
Chain scheduling based on JEC & GFac Config
GFac-Core
Application Data
Host Data
inMessageContext
outMessageContext
Security Context
Original Request

Airavata Client API's invocations travers through jobs/workflows input binding inputs and executing through the process of Resource Scheduling, Data Staging and finally invoke Providers and Handlers. Gateway developers can configure the execution order of their handlers using a simple XML configuration. The framework provides pluggable ***Global***; ***Provider-specific***; ***Application-specific***; ***In-flow*** and ***Out-flow*** handlers.

Global InHandlers
Application Specific InHandlers
Provider Specific InHandlers
Provider Logic
Global OutHandlers
Application Specific OutHandlers
Provider Specific OutHandlers

We plan to build upon the architecture and investigate third party development of providers and handlers and further loosen the coupling by providing deployment bundles.

Acknowledgements

The authors would like to thank the Apache Airavata project management committee for continued development of a open community science gateway framework. The Apache Airavata project is fostered by OGCE Collaboration - Open Gateway Computing Environments through NSF Grant No: ACI 1032742. Further information:

http://airavata.apache.org