JSAGA adaptors

14/02/2012

Sylvain Reynaud

Lionel Schwarz

dapnia

ceea

saclay

CNRS

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

The JSAGA implementation

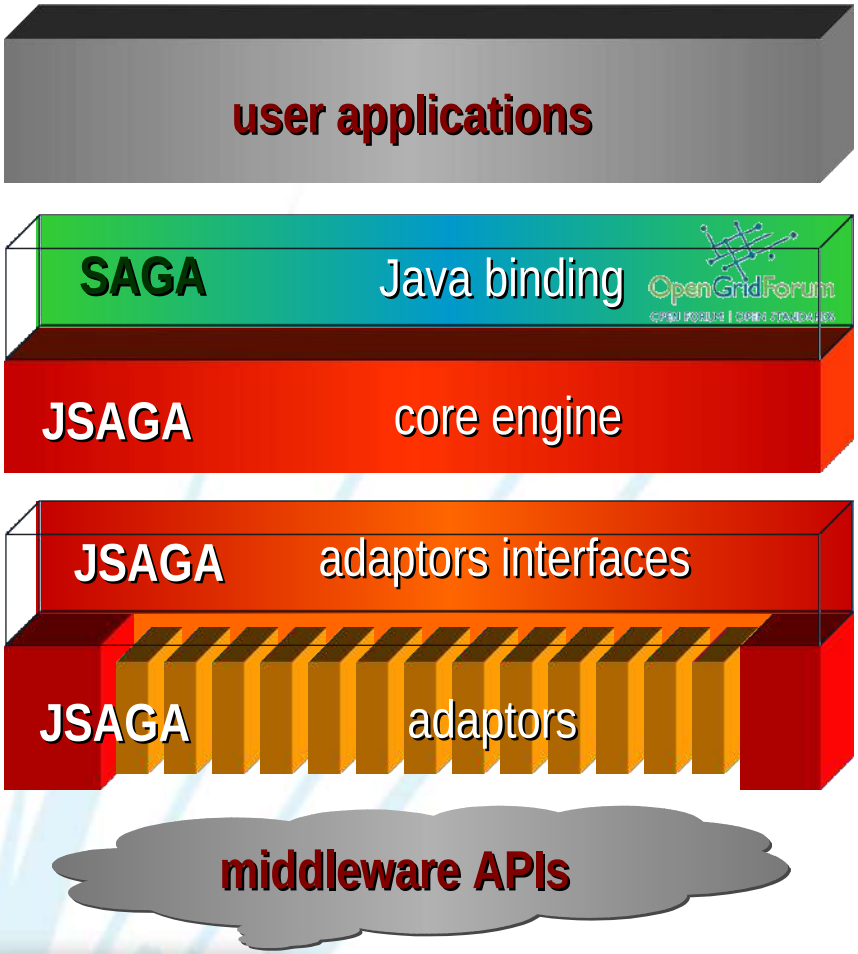
- a **J**ava implementation of the **SAGA** specification

- Focuses on
  - uniform access to middlewares.. (beyond what they natively support)
  - ease of extension…… } (thanks to the design of adaptor interfaces)
  - efficiency and scalability……
  - advanced configuration …… (contexts bound to remote system)
  - operating-system independence (tested on)
- Under LGPL license

Layered software architecture

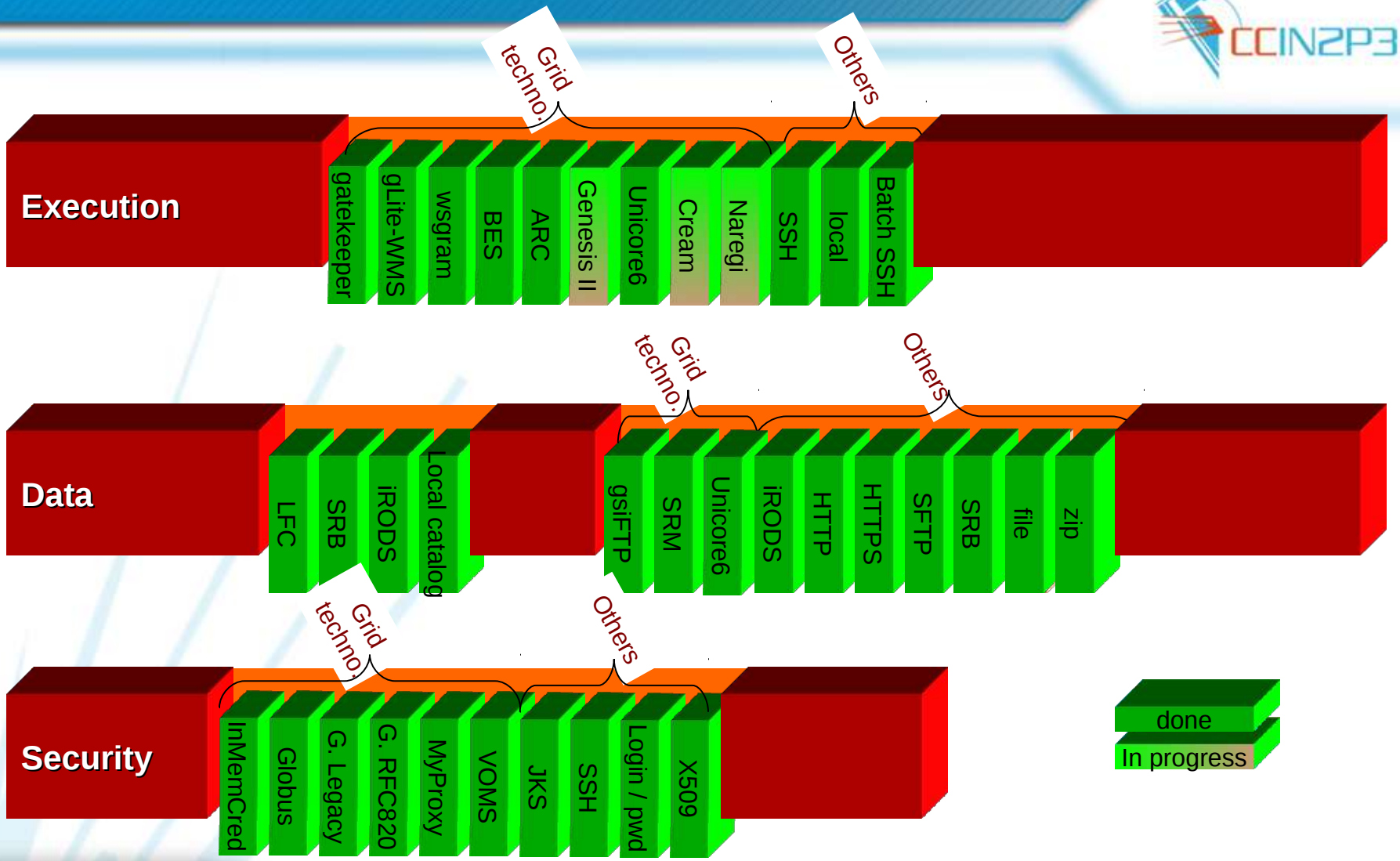
interface

implementation



- Used by end users
  - web portals, GUI, CLI…
- Used by application developers
  - a single uniform interface
  - object oriented, high-level
- Implemented by adaptors
  - each "way" to implement a given feature has 1 interface
  - service oriented, low-level
- Implemented by middleware
  - each technology has its own interfaces

Supported technologies

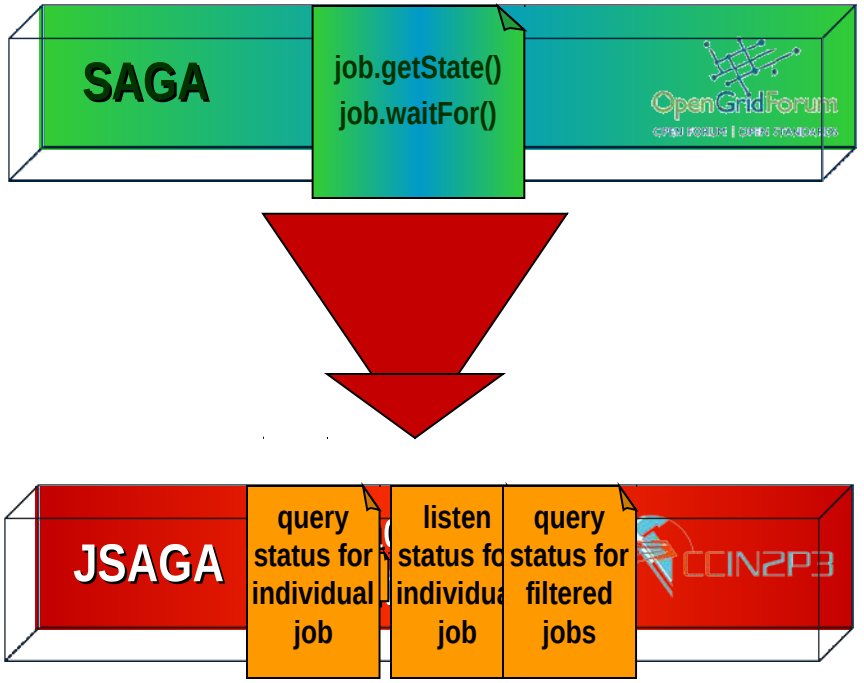


Designing your adaptor

- What is the purpose of your adaptor ?
  - Manage job → job adaptor
  - Manage data → data adaptor
  - Manage security credential → security adaptor
- What type of security credential does your adaptor need ?
  - If already exists in JSAGA, simply use it
  - Otherwise, develop a new security adaptor
    - Your adaptor can inherit from other security adaptors
- What do the targeted middleware support ?
  - You have to match middleware features and JSAGA interfaces
    - http://grid.in2p3.fr/software/jsaga-dev/contributors-guide.html

Designing you job adaptor

- Which job adaptor interfaces to implement? It depends on
  - what service you want to provide (submission, monitoring, information, data staging...)
  - what is supported by the targeted middleware
- Example: ways to **monitor a job**
  - Choose mode (poll vs notification)
  - Choose granularity (single job, list of jobs, list of filtered jobs)



Contributor's guide

Developing a job monitor adaptor

A job monitor adaptor **must** implement the JobMonitorAdaptor interface.

A job monitor adaptor **must** implement at least one of the monitoring interfaces. It **may** implement several if they are natively supported by the targeted scheduler. Monitoring interfaces are:

- QueryIndividualJob: Query status for a single job
- QueryListJob: Query status for a list of jobs
- QueryFilteredJob: Query status for jobs maching a filter expression
- ListenIndividualJob: Listen to status changes for a single job
- ListenFilteredJob: Listen to status changes for jobs matching a filter expression

Monitoring interfaces create job status objects. A job status object **must** extend the JobStatus abstract class.

A job monitor adaptor **may** implement the JobInfoAdaptor optional interface.

A job monitor adaptor **may** implement the ListableJobAdaptor interface in order to list user jobs known by the targeted scheduler service.

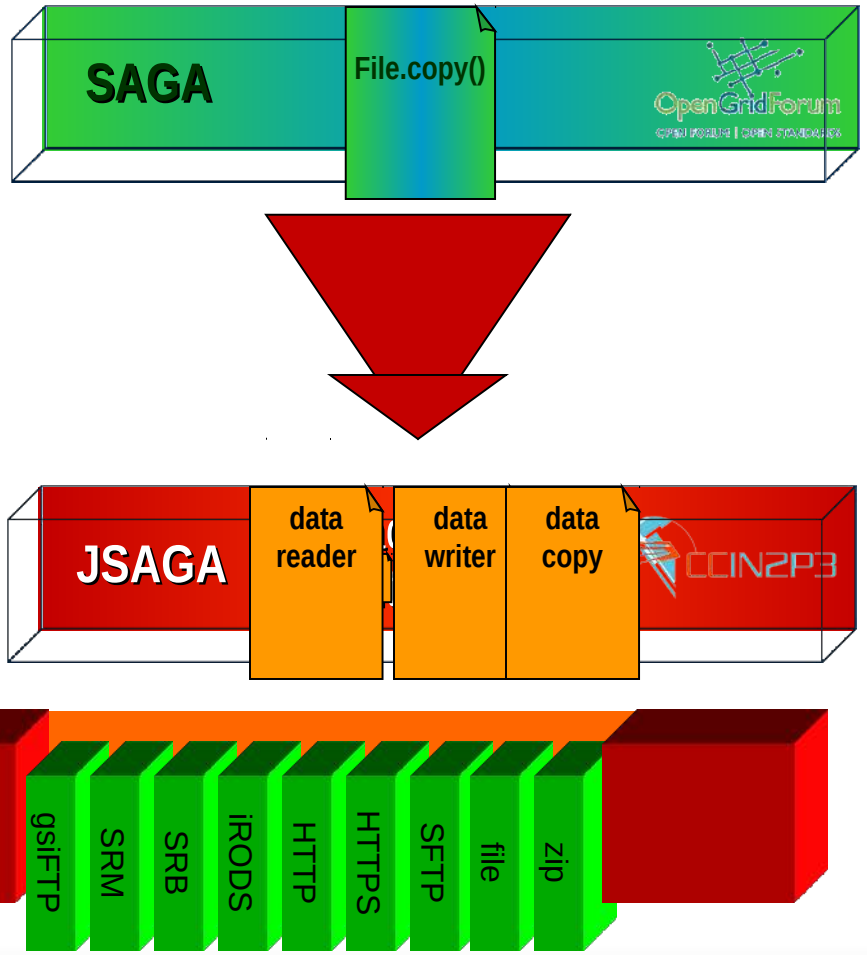
Copy-paste required methods to your adaptor class, and implement them.

```
    // methods of interface Adaptor
    public String getType() {
        return null; //todo: this method MUST be implemented!
    }
    public Usage getUsage() {
        return null; //todo: this method MUST be implemented!
    }
    public Default[] getDefaults(Map attributes) throws IncorrectStateException {
        return null; //todo: this method MUST be implemented!
    }

    // methods of interface ClientAdaptor
    public Class[] getSupportedSecurityCredentialClasses() {
        return null; //todo: this method MUST be implemented!
    }
```

Designing your data adaptor

- Optional interfaces for optimization
- Example: ways to **copy a file**
  - data read/write methods (stream methods or get/put methods)
  - data copy (e.g. enable third-party transfer)
  - delegated transfer



Implementing your adaptor

- JDK>=1.5 is needed
- Maven 2.2.1 as build automation system
  - JSAGA archetype to create skeleton for your project
  - external libraries : better if provided in POM, otherwise they will be located in the JSAGA « external » subproject
- Implement appropriate adaptor interfaces
  - http://grid.in2p3.fr/software/jsaga-dev/contributors-guide.html
  - use other adaptors as example
- Contributors Frequently Asked Questions
  - http://grid.in2p3.fr/software/jsaga-dev/contributors-faq.html

Testing your adaptor

- Standard SAGA integration tests suite
- For your adaptor, you simply:
  - Disable all standard tests for features that your adaptor does not support
  - Configure test files
    - jsaga-default-context.xml: configuration of adaptor and its security context
    - log4j.properties: log configuration for test-suite
    - saga-test.properties: test-suite configuration (URLs and test parameters)
  - Run your tests in your IDE or with Maven CLI

JSAGA community

- Composed of:
  - Core developers (and most of adaptors)
  - Contributors (mainly specific adaptors)
  - Users (developers of high-level apps sit on top of JSAGA)
- Integrate your adaptor into JSAGA project
  - Using GIT contributors banches
  - Or sending patches that the JSAGA team will integrate
- Exchange with other contributors and users
  - In the forum : https://forge.in2p3.fr/projects/jsaga/boards
  - In the Wiki: https://forge.in2p3.fr/projects/jsaga/wiki

Conclusion

- For JSAGA to support access to a remote system :
  - First check existing adaptors (security, jobs, data)
  - Develop your own adaptors using inheritance when it is possible
- When developing a new adaptor, your job is to:
  - Design and implement supported features
  - Configure your test environment (security, adaptors...)
  - Check that your adaptor works fine with standard SAGA test suite
  - Document it
- You do not have to :
  - Worry about integration of your adaptor into the JSAGA engine