Open Grid Forum

Catania March 02, 2009

QoS management in Grids

Di Stefano A., Morana G., Zito D.

Dep. of Computer Science and Telecommunication Engineering

Engineering Faculty - Catania University

Catania University

Cometa Consortium

QoS management in Grids

The concept of QoS is strictly related to the **perception that the users** have of the service fruition:

- Execution time
- Maximum queue waiting time
- Response time
- Cost

High level QoS management

- Manages the interaction with the user, collecting its requirements and translating them in specific requests of underlying resources.

Low level QoS management

- Manages the interaction with the underlying system (hardware and software), coordinating the resources needed to satisfy the user requests

QoS management in Grids

A user request as:

I want that service X is executed in Y minutes

implies that the system is able to:

(1) check the availability of software resources for service X

(2) check the availability of hardware resources for service X

(3) identify among the resources the ones able to satisfy the constrain "Y min"

(4) if impossible to satisfy, re-negotiate the Y value

(5) if impossible to satisfy, preempt the current running task

(6) if necessary, reserve resources only for the execution of Service X

(7) monitoring the services in order to guarantee the given deadline

QoS management in Grids

A user request as:

I want that service X is executed in Y minutes

implies that the system is able to:

(1) check the availability of software resources for service X

(2) check the availability of hardware resources for service X

(3) identify among the resources the ones able to satisfy the constrain “Y min”

(4) if impossible to satisfy, re-negotiate the Y value

(5) if impossible to satisfy, preempt the current running task

(6) if necessary, reserve resources only for the execution of Service X

(7) monitoring the services in order to guarantee the given deadline

Information Service

Workload Management System

QoS management in Grids

A user request as:

I want that service X is executed in Y minutes

implies that the system is able to:

(1) check the availability of software resources for service X

(2) check the availability of hardware resources for service X

(3) identify among the resources the ones able to satisfy the constrain "Y min"

(4) if impossible to satisfy, re-negotiate the Y value

(5) if impossible to satisfy, preempt the current running task

(6) if necessary, reserve resources only for the execution of Service X

Manager of Resources Reservation

(7) monitoring the services in order to guarantee the given deadline

Low Level QoS Management

QoS management in Grids

A user request as:

I want that service X is executed in Y minutes

implies that the system is able to:

(1) check the availability of software resources for service X

(2) check the availability of hardware resources for service X

(3) identify among the resources the ones able to satisfy the constrain "Y min"

(4) if impossible to satisfy, re-negotiate the Y value

(5) if impossible to satisfy, preempt the current running task

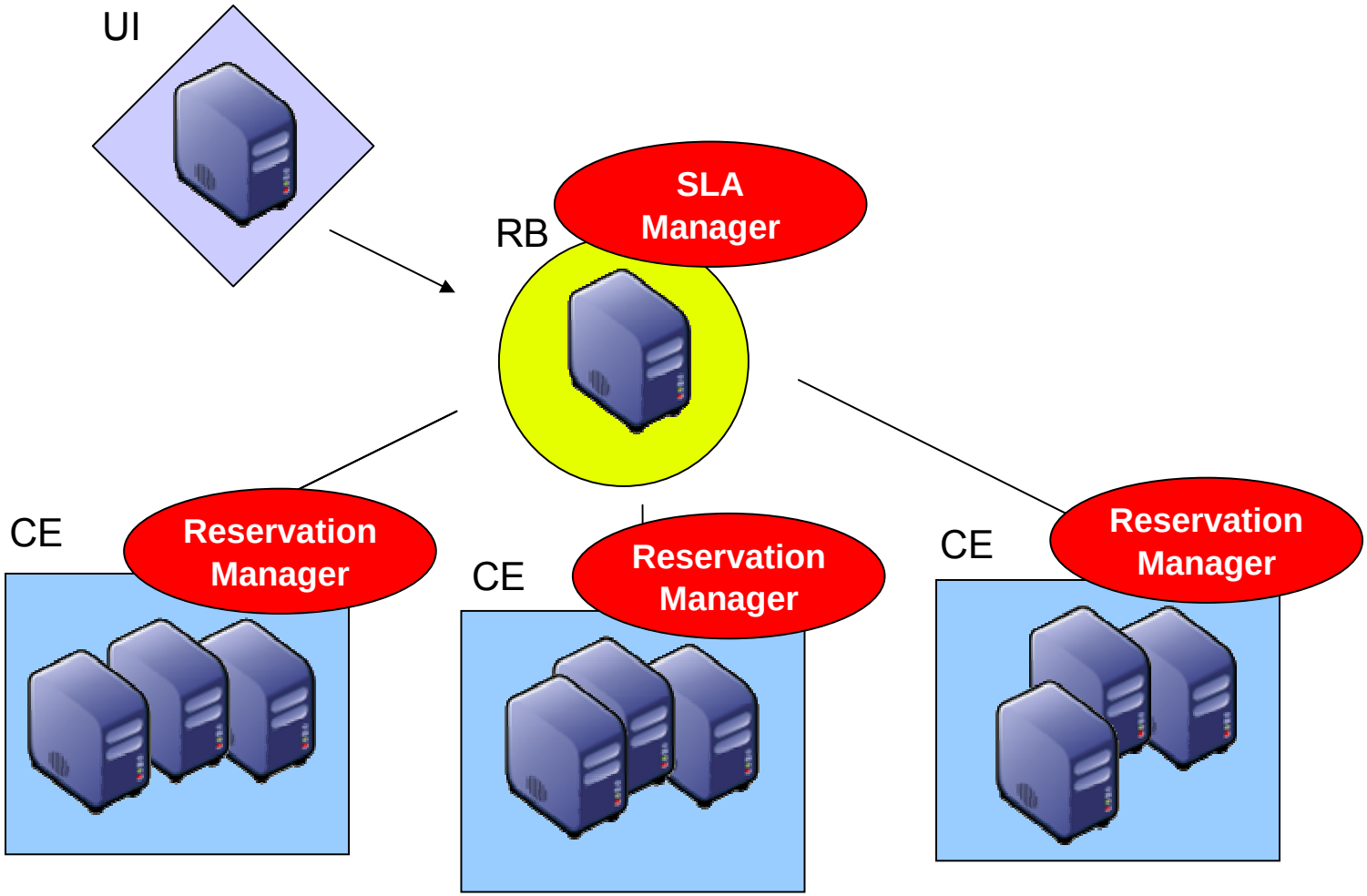
(6) if necessary, reserve resources only for the execution of Service X

(7) monitoring the services in order to guarantee the given deadline

Manager of Services Agreements

High Level QoS Management

Deploy on gLite



Resource Reservation

- Today, the majority of the grids provide only best effort services and are not able to assure the "QoS guaranteed" services execution (based on resources reservation)

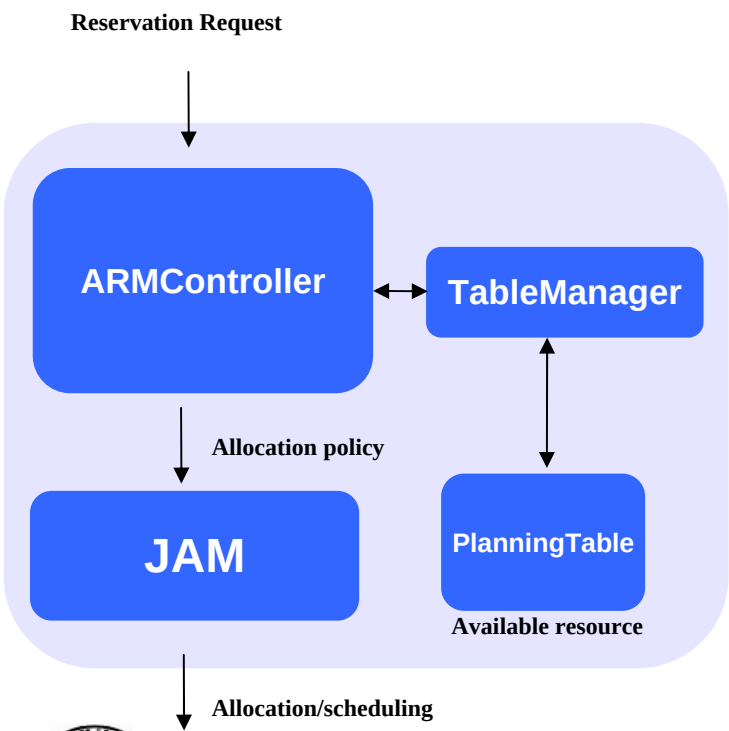
- Furthermore, the **existing solution for resources reservation (**job submission service):
  - allow "**immediate**" but not "advanced" reservation
  - reserve the **whole computational resource**, not portions of it.
  - **bind** the **reservation** policy to the **user** and not to the job

Those solutions are static, not very flexible

and designed *ad hoc* for specific applications

Reservation Pattern (Low Level)

1. Decoupling of the phases research-selection-reservation of the resource
2. Handling of "time" dimension
3. Resources Virtualization



ARMController
Starts, monitoring, checks and stops the reservations

JAM
Allocates the resources needed for the reservations

PlanningTable
Maintains the info related to the resources availability

TableManager
Update the PlanningTable entries

Platform LSF & EGO

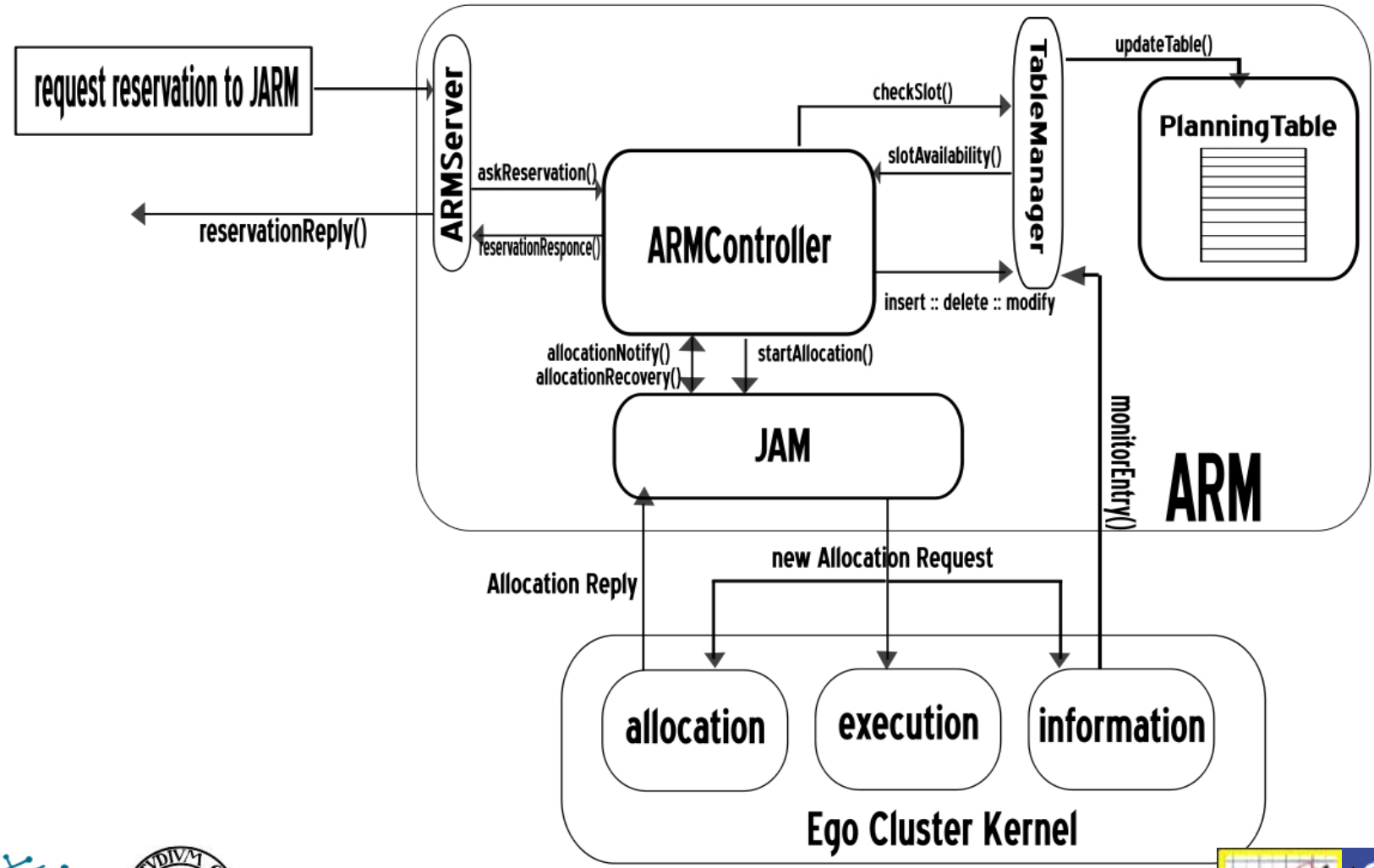
The Production Grid of PI2S2 project has:

- **gLite (EGEE)** as middleware
- **LSF (Platform)** as CE scheduler

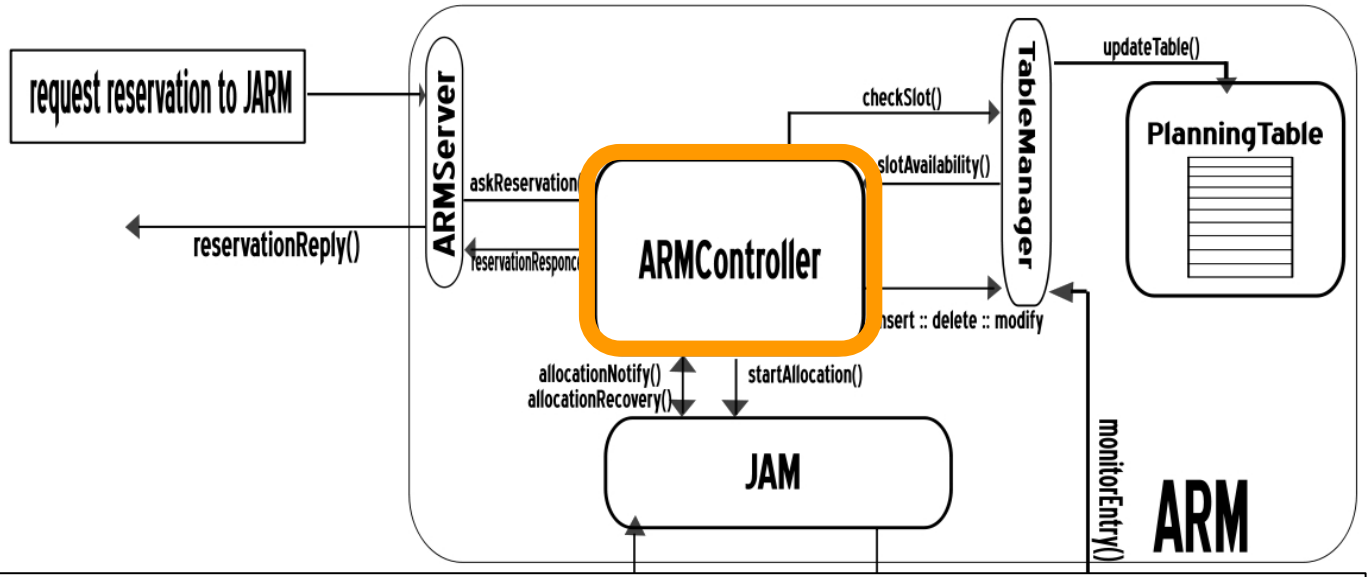
LSF → EGO Cluster kernel:

- provides the virtualization of the underlying computational resources
- provides three basic services: information, allocation and execution.

Reservation Pattern (gLite)



ARM Controller



ARMController:

(1) receives the reservation requests

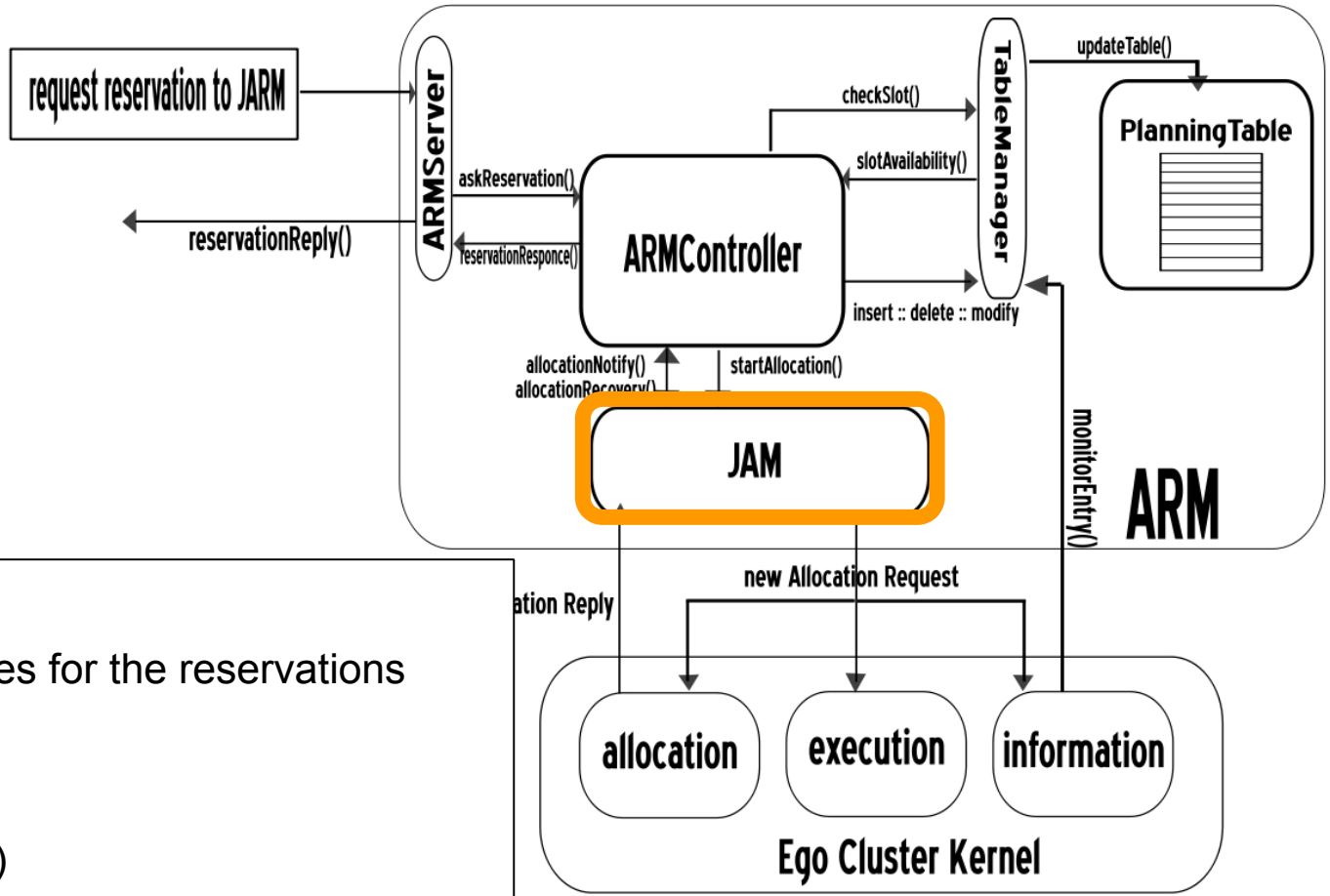
(2) Starts, monitors and stops the:

- immediate reservation (allocation)
- advance reservation
- oscillating reservation

(3) Manages the PlanningTable through TableManager

(4) Deletes the pending reservations (not confirmed after a given time)

Job Allocation Manager



JAM:

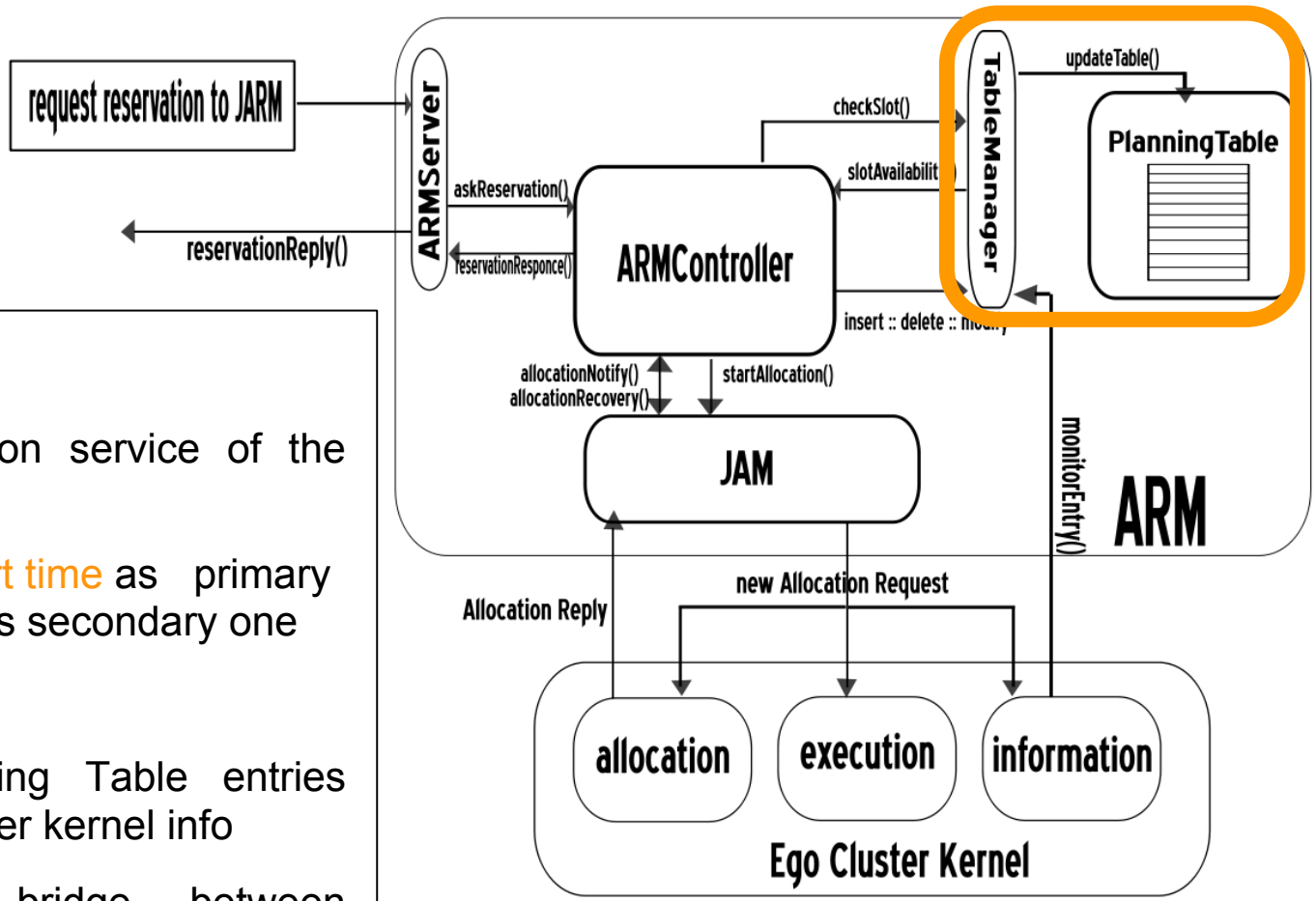
(1) Allocates the resources for the reservations

(2) Three thread:

- Resource
- Work (execution)
- Monitor

(3) Represent the EGO cluster kernel interface

Planning Table & TableManager



Planning Table:

(1) It is the information service of the reservation system

(2) hashtable with start time as primary key and end time as secondary one

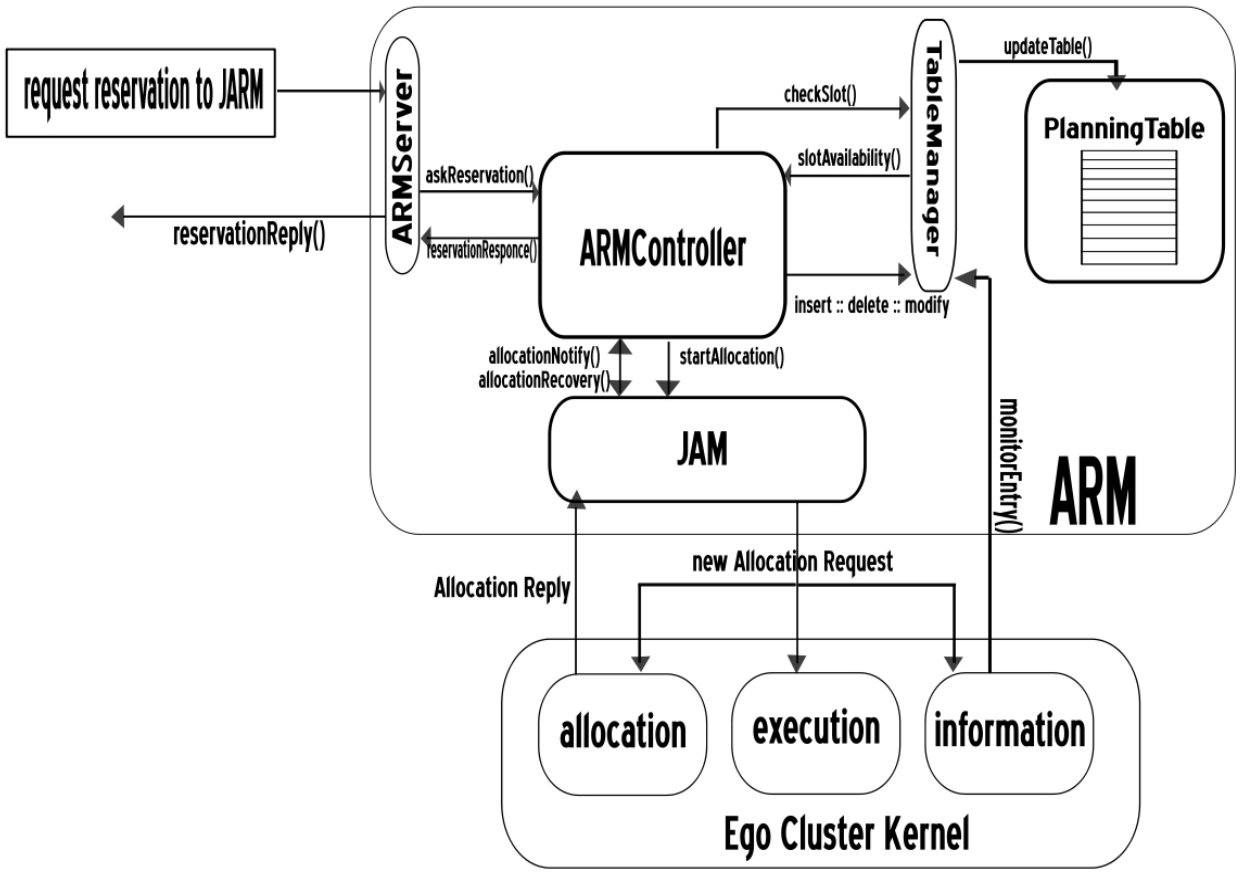
TableManager:

(1) update the Panning Table entries with the EGO cluster kernel info

(2) represents the bridge between planning table, ARM and EGO

Collaboration Diagram

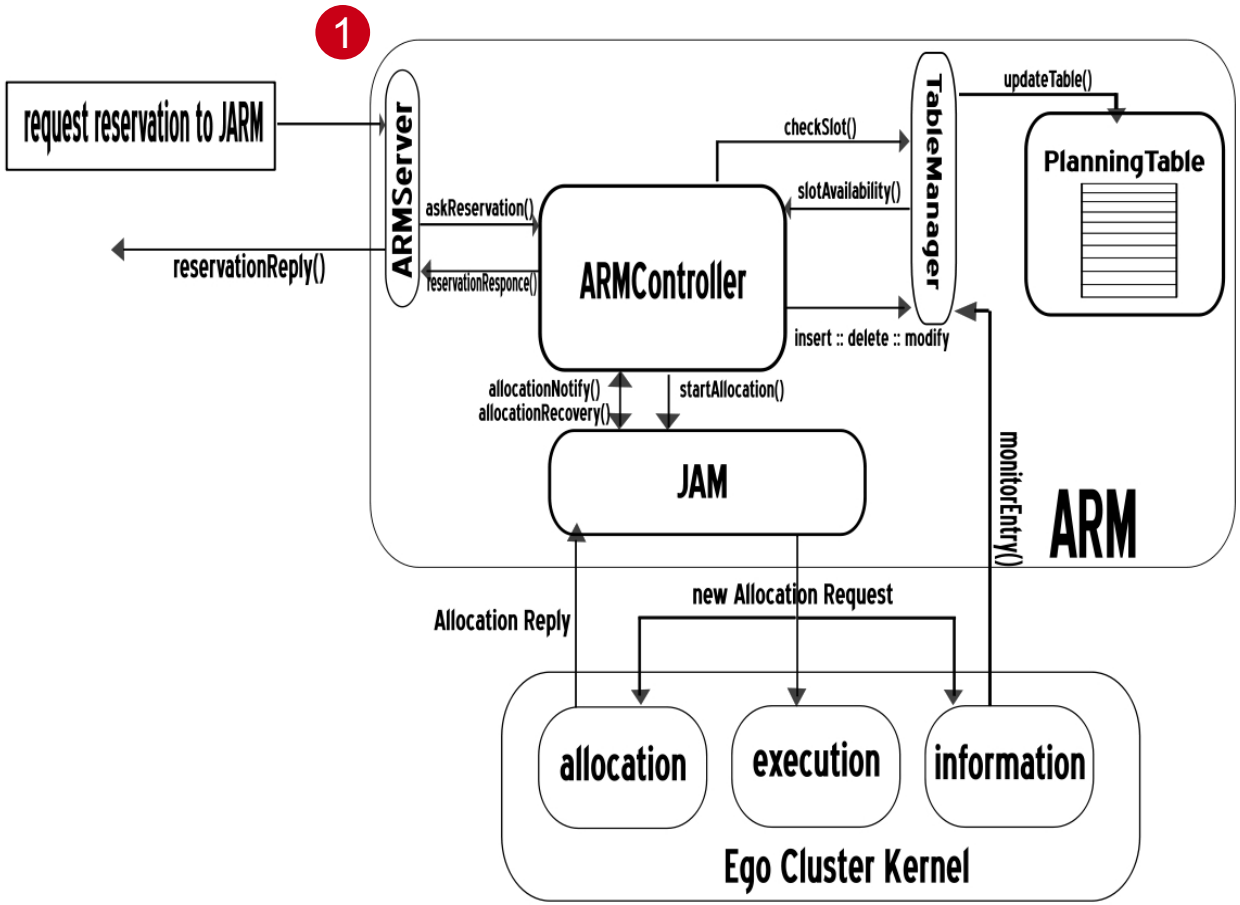
Use case: Incoming Reservation Request



Collaboration Diagram

Use case: Incoming Reservation Request

Reservation Request

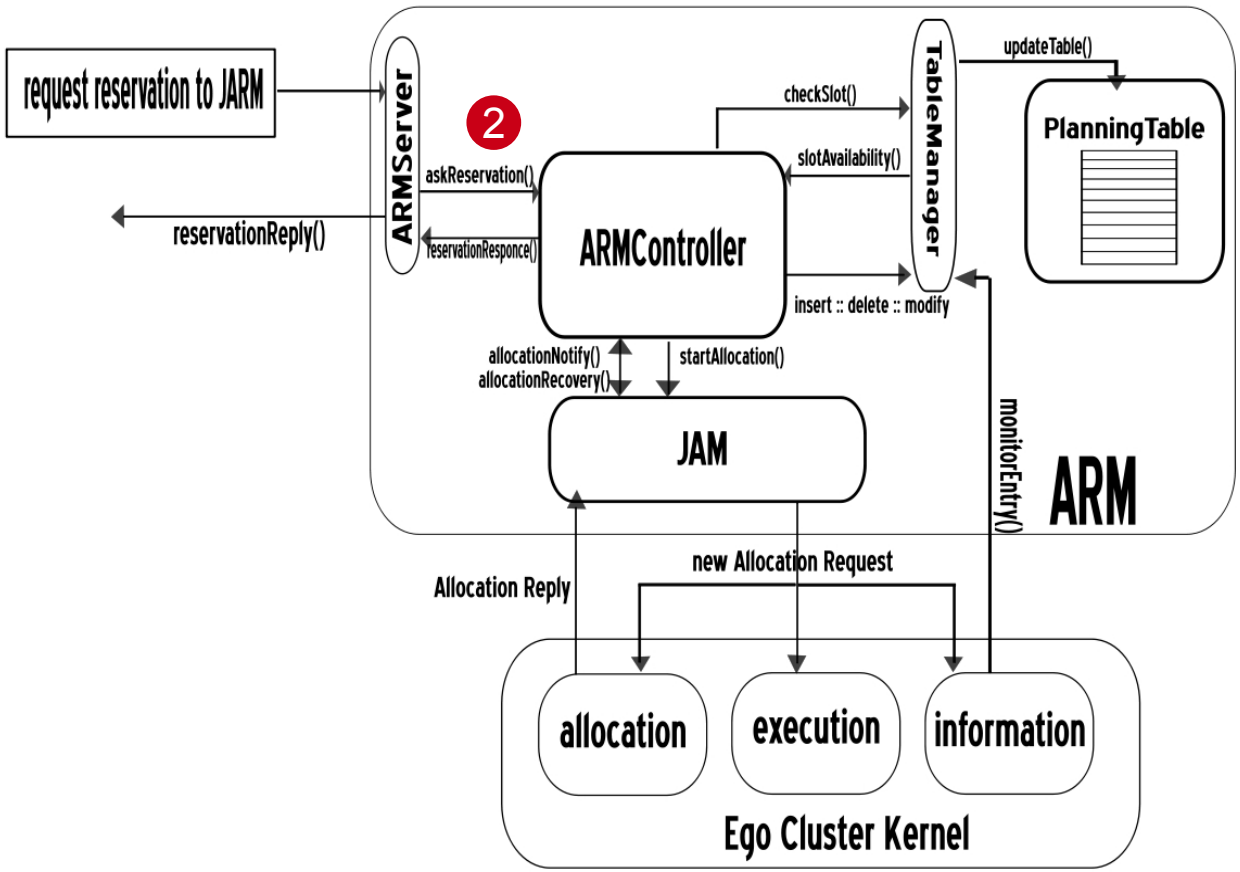


Collaboration Diagram

Use case: Incoming Reservation Request

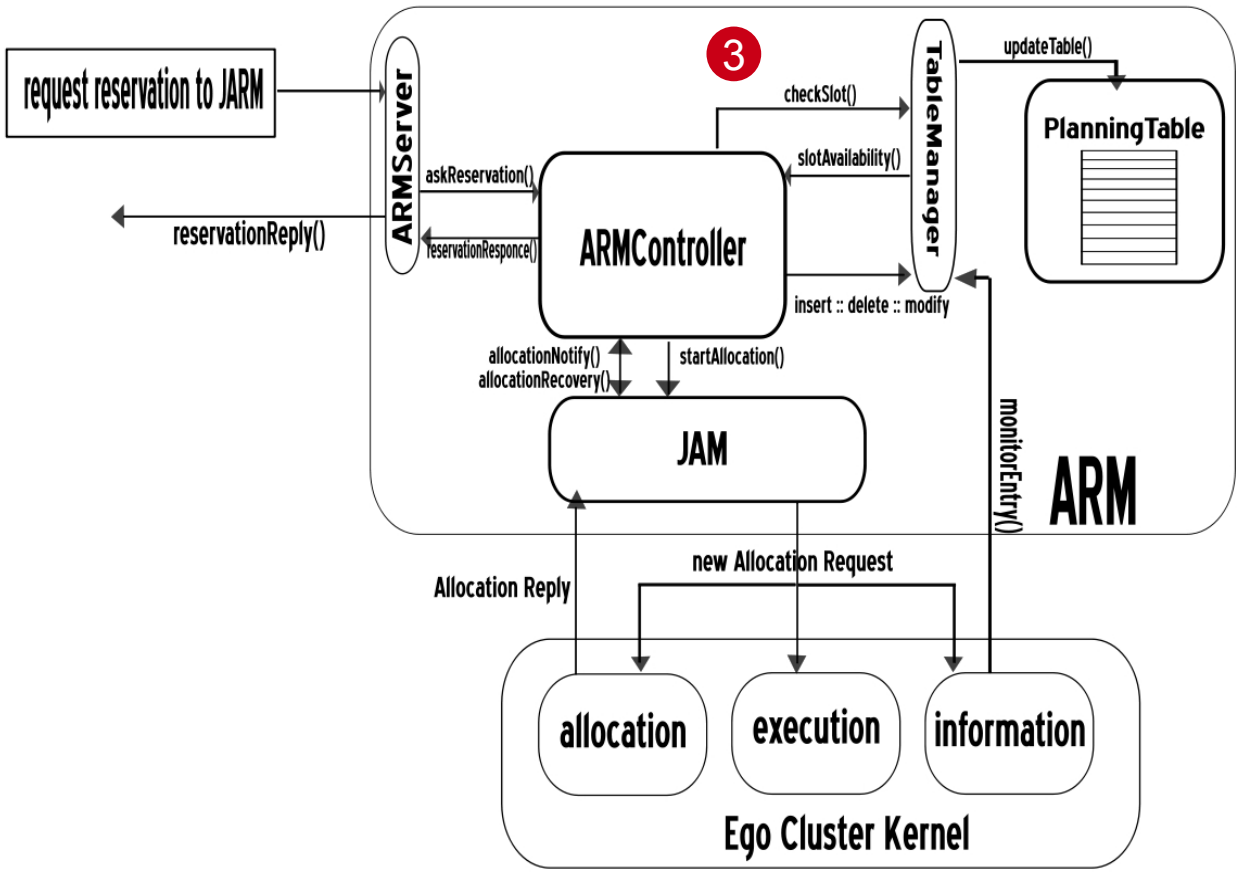
Reservation Request

JDL parsing



Collaboration Diagram

Use case: Incoming Reservation Request



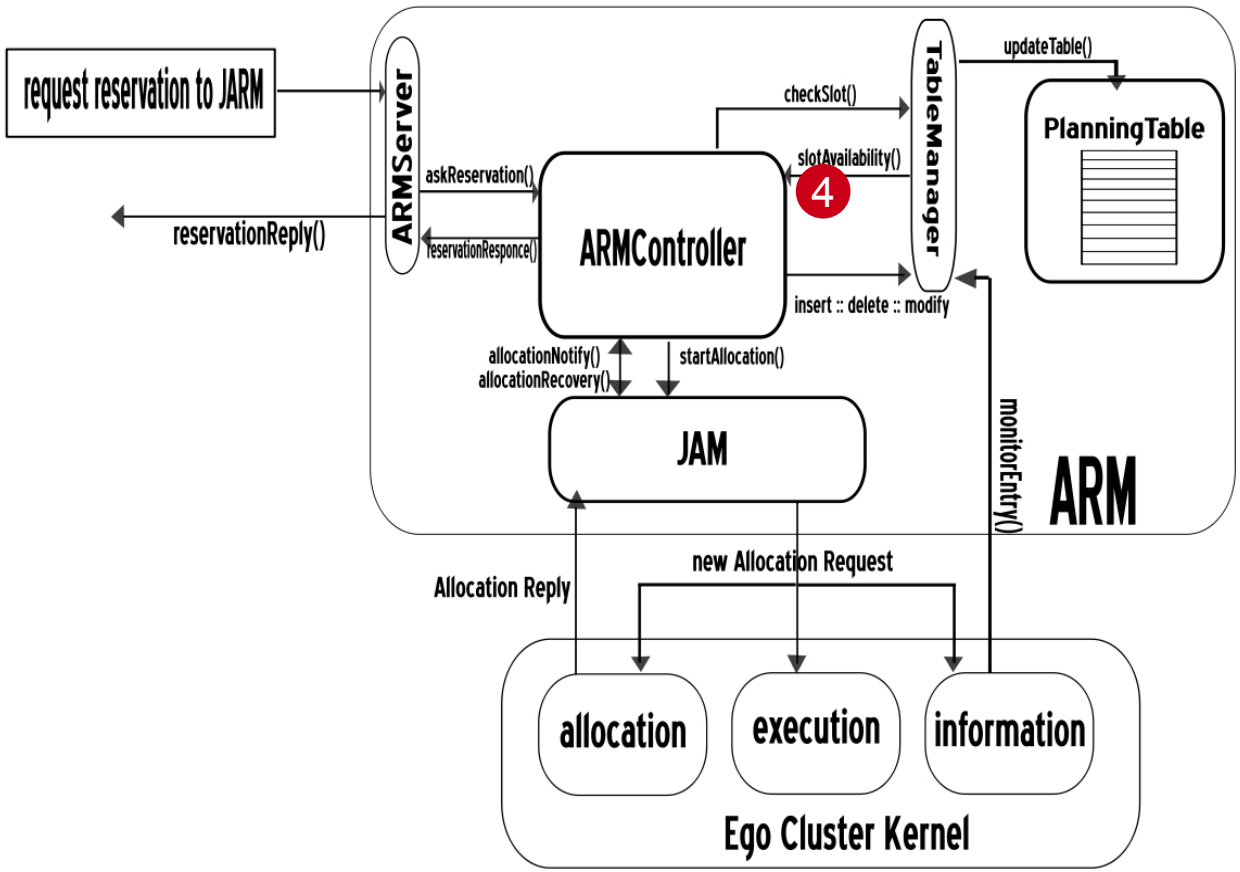
Reservation Request

JDL parsing

Planning Table interrogation

Collaboration Diagram

Use case: Incoming Reservation Request



Reservation Request

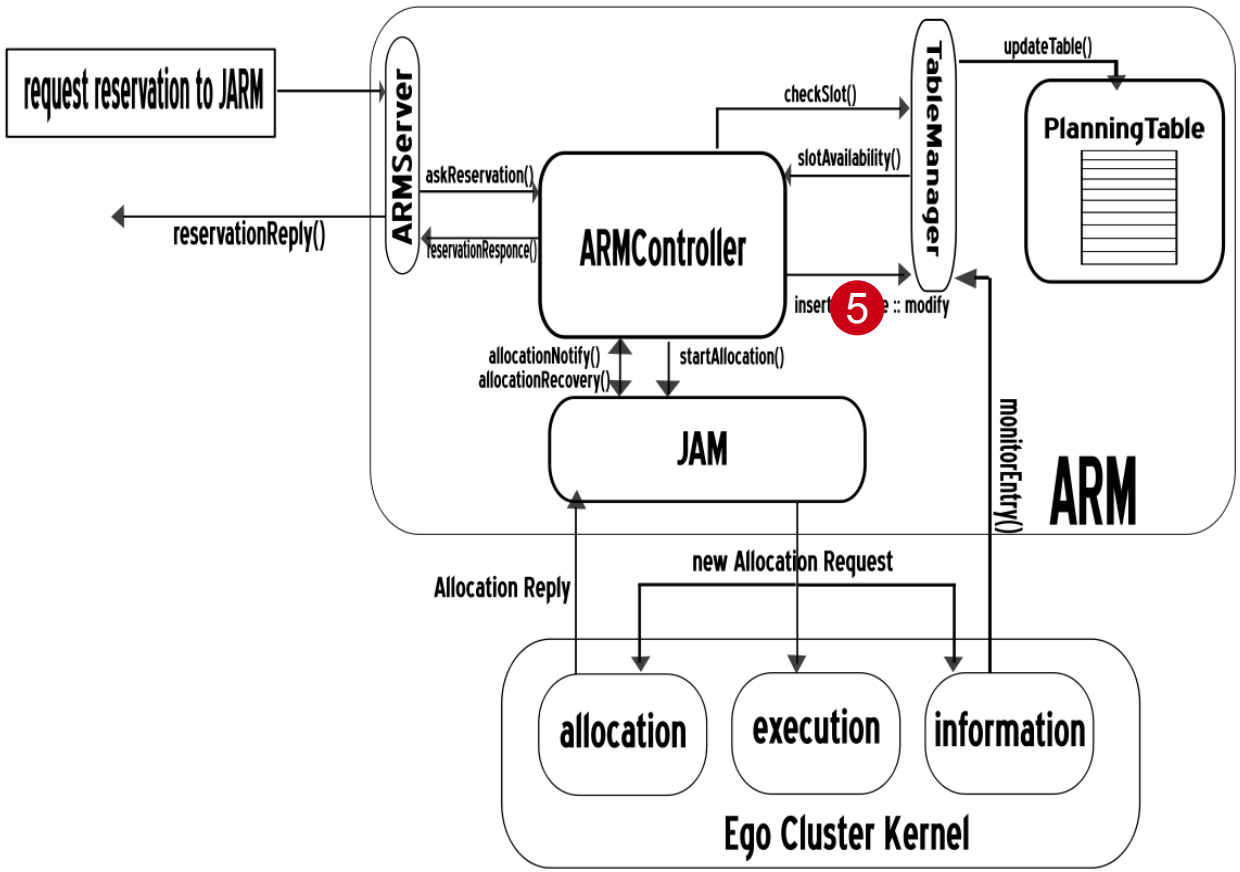
JDL parsing

Planning Table interrogation

List of free slots or other reservation notification

Collaboration Diagram

Use case: Incoming Reservation Request



Reservation Request

JDL parsing

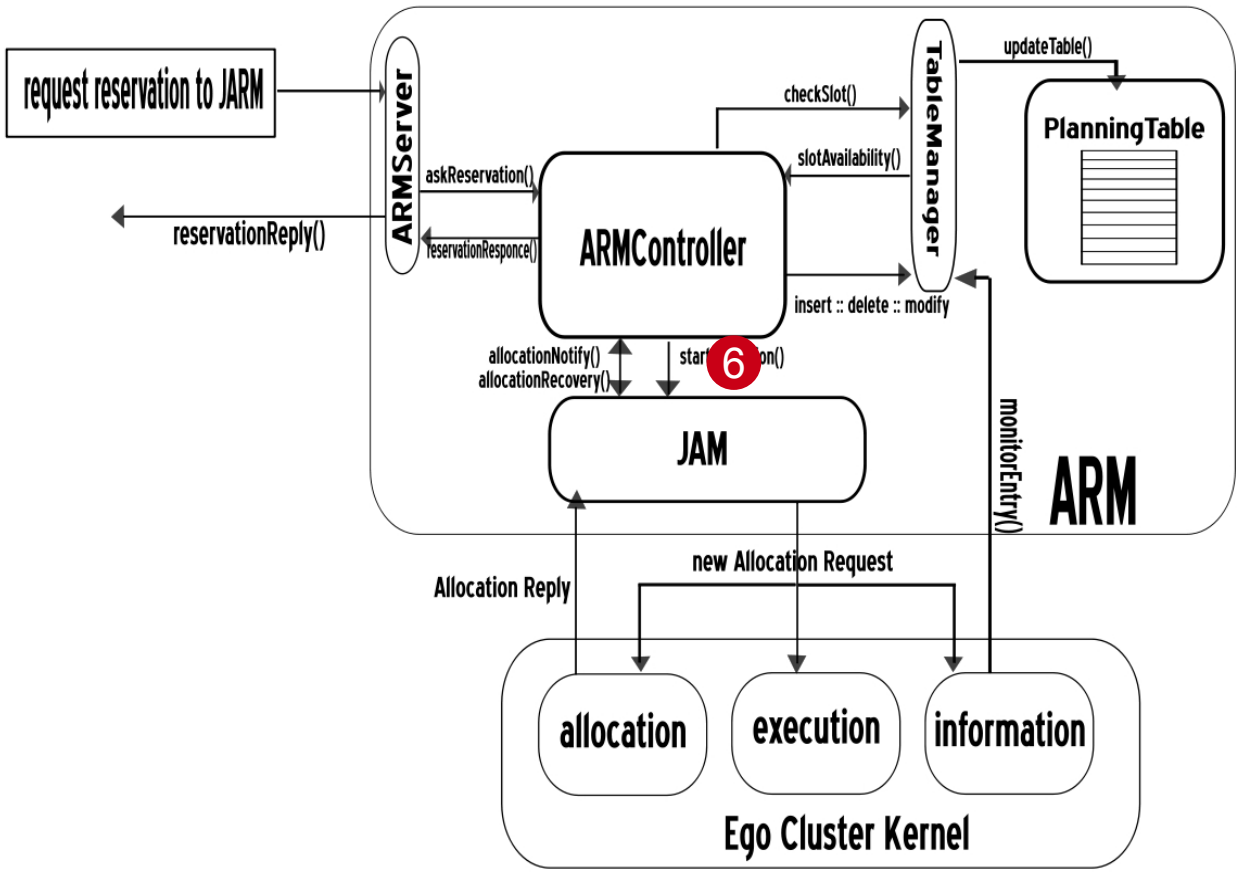
Planning Table interrogation

List of free slots or other reservation notification

Resource reservation on PlanningTable

Collaboration Diagram

Use case: Incoming Reservation Request



Reservation Request

JDL parsing

Planning Table interrogation

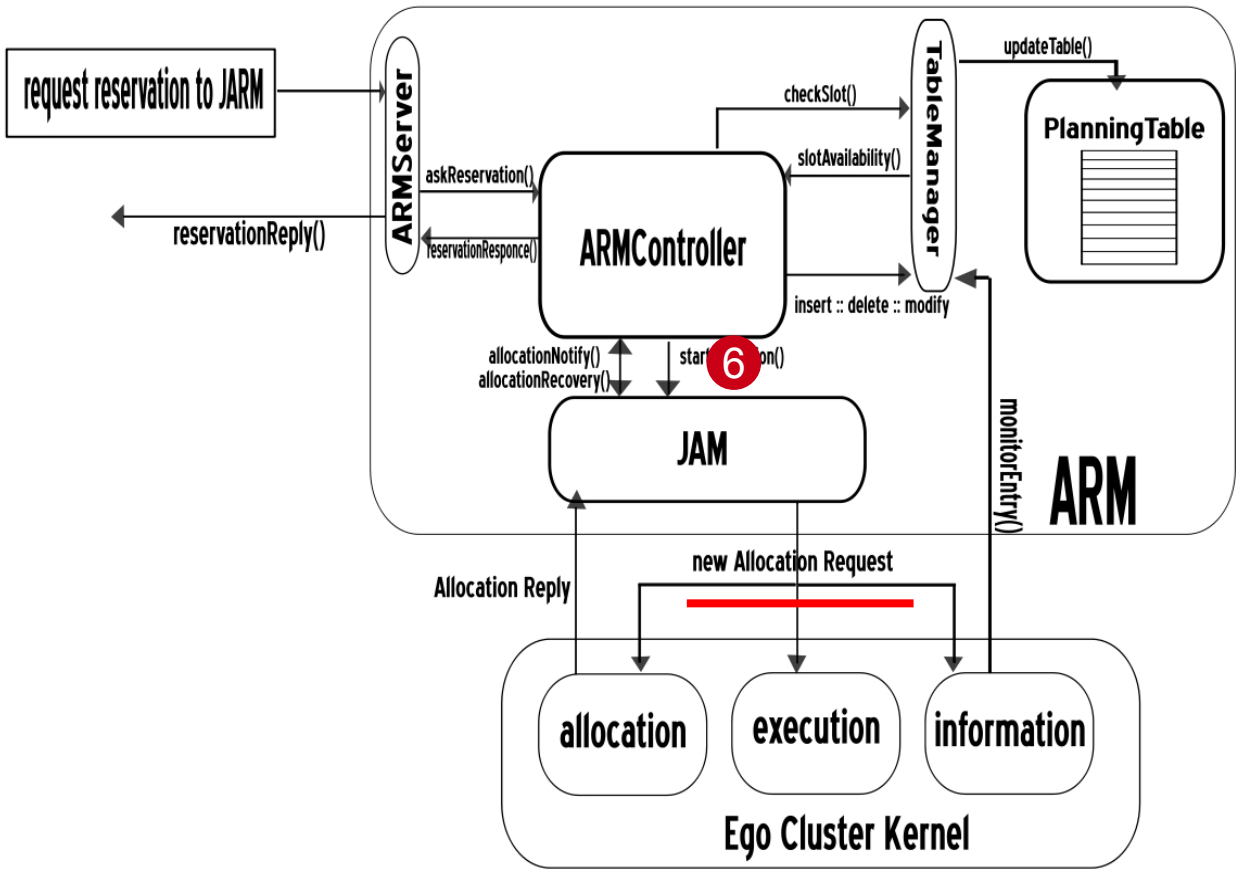
List of free slots or other reservation notification

Resource reservation on PlanningTable

JAM notification

Collaboration Diagram

Use case: Incoming Reservation Request



Reservation Request

JDL parsing

Planning Table interrogation

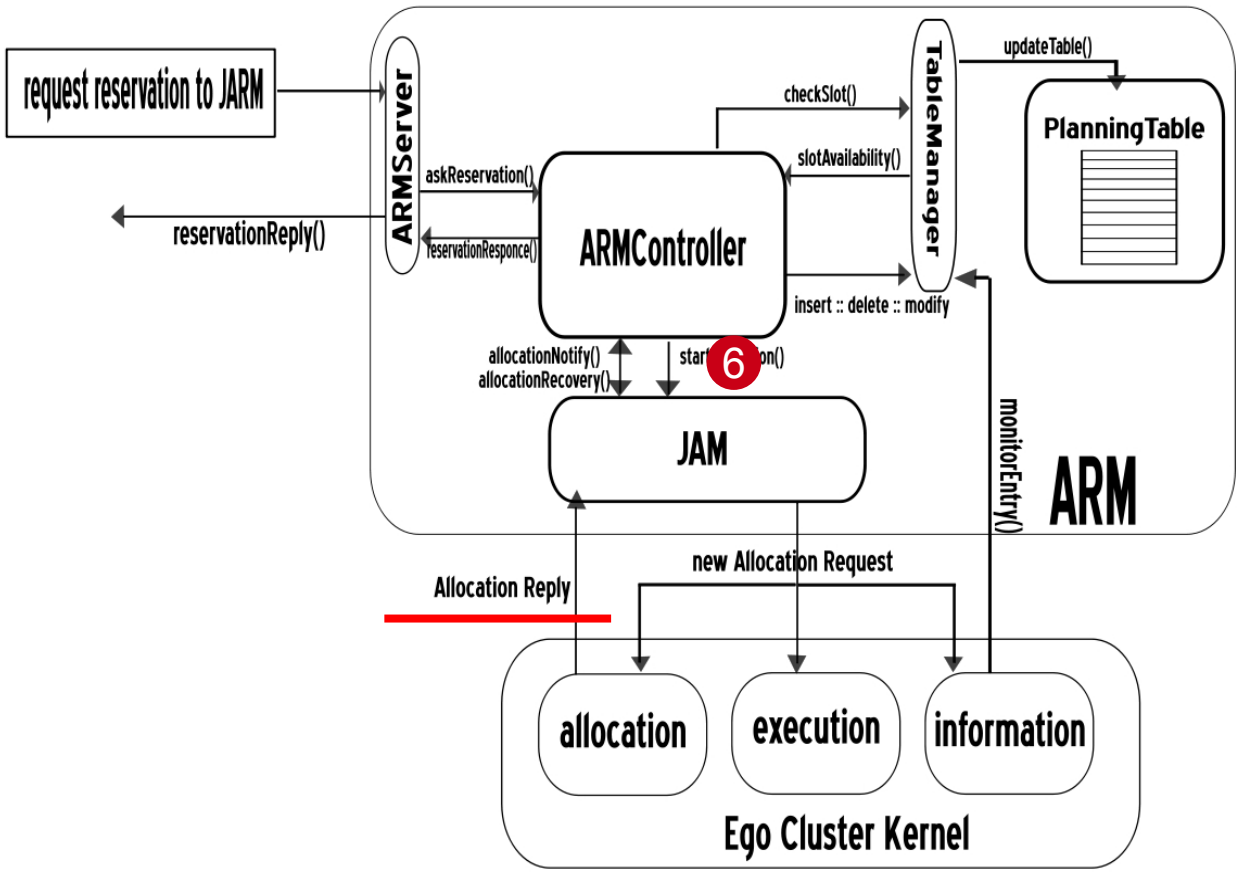
List of free slots or other reservation notification

Resource reservation on PlanningTable

JAM notification

Collaboration Diagram

Use case: Incoming Reservation Request



Reservation Request

JDL parsing

Planning Table interrogation

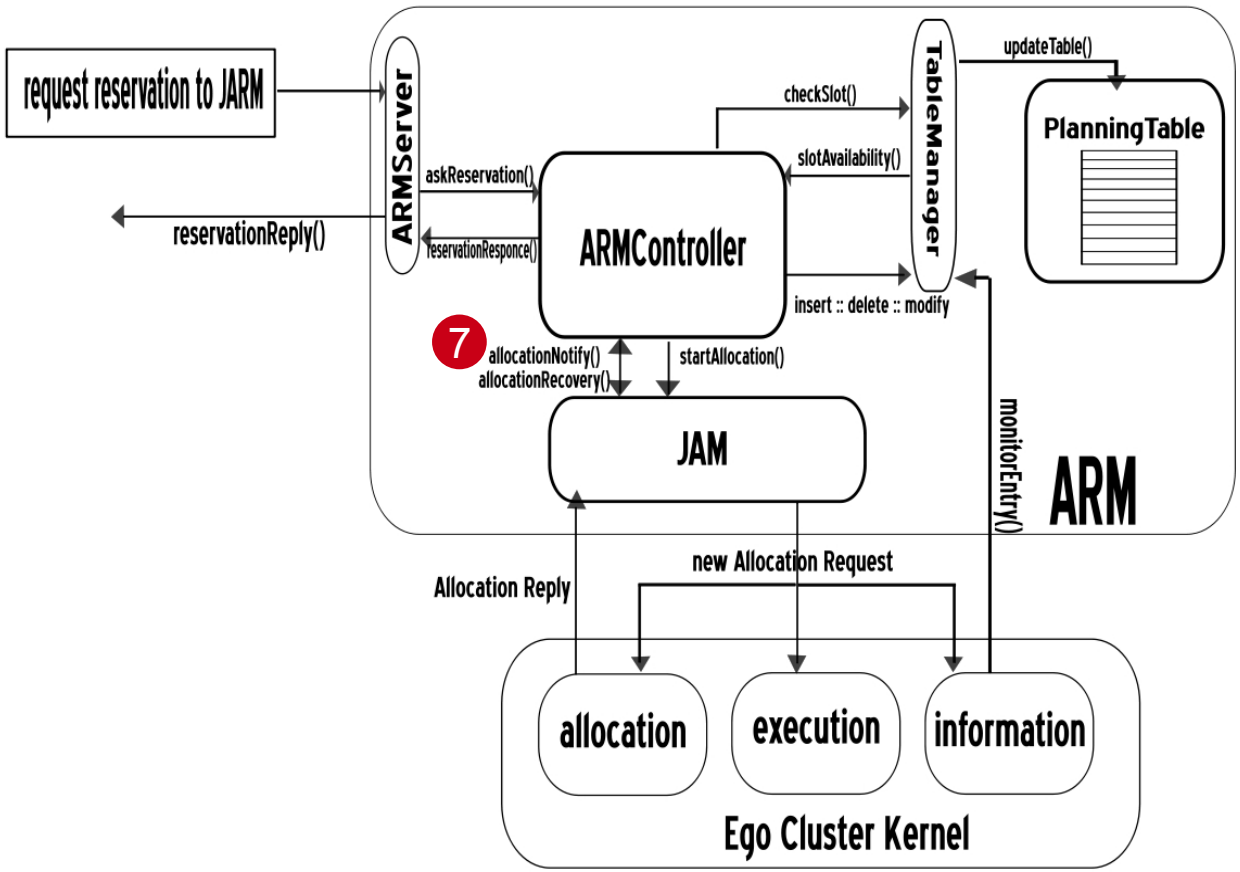
List of free slots or other reservation notification

Resource reservation on PlanningTable

JAM notification

Collaboration Diagram

Use case: Incoming Reservation Request



Reservation Request

JDL parsing

Planning Table interrogation

List of free slots or other reservation notification

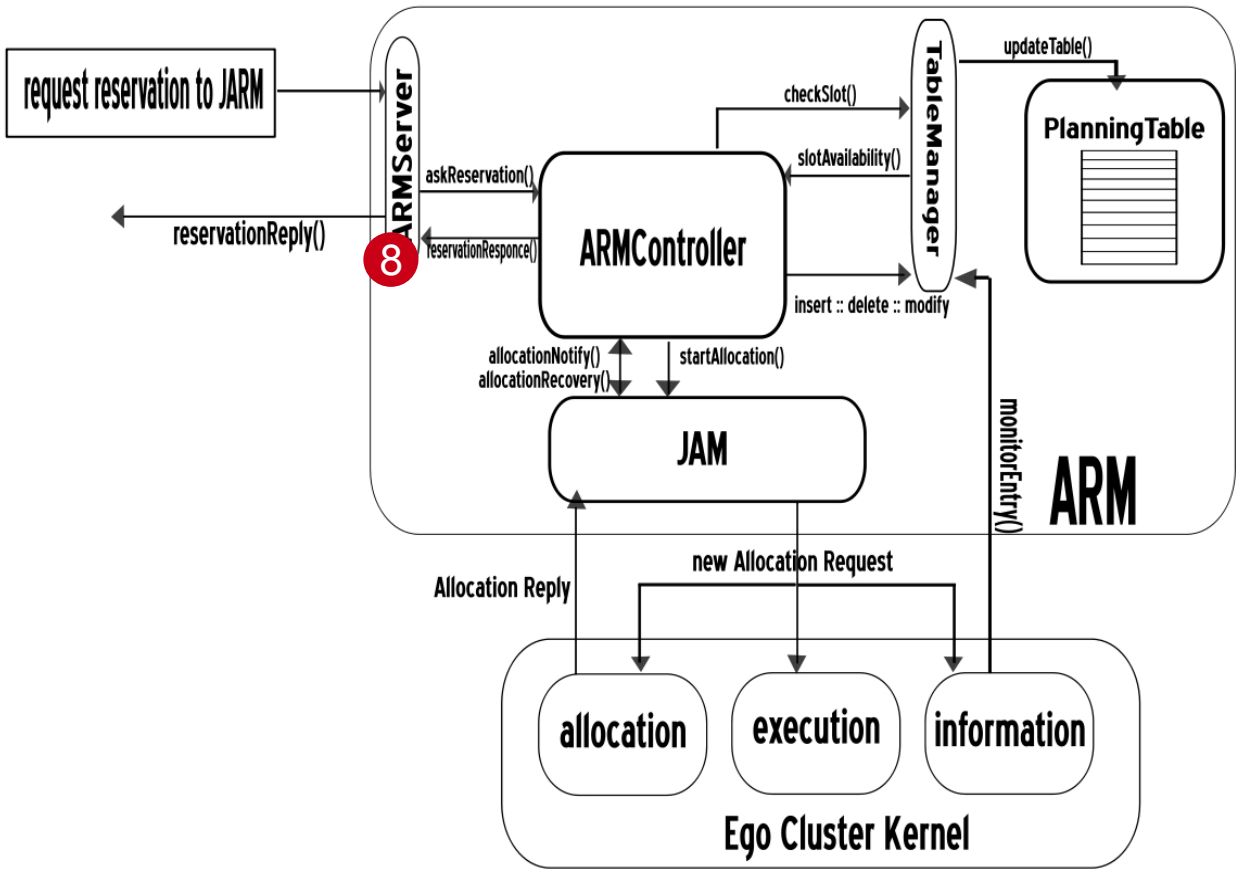
Resource reservation on PlanningTable

JAM notification

Notification of happen reservation

Collaboration Diagram

Use case: Incoming Reservation Request



Reservation Request

JDL parsing

Planning Table interrogation

List of free slots or other reservation notification

Resource reservation on PlanningTable

JAM notification

Notification of happen reservation

Reservation Confirm

Reservation Pattern (Advantages)

- Allows to bind the reservation to a job or to a jobs group
- Allows the advanced resource reservation
- The resources virtualization allows a more effective resources control
- The decoupling of components allows the use of different scheduling policies for resources management

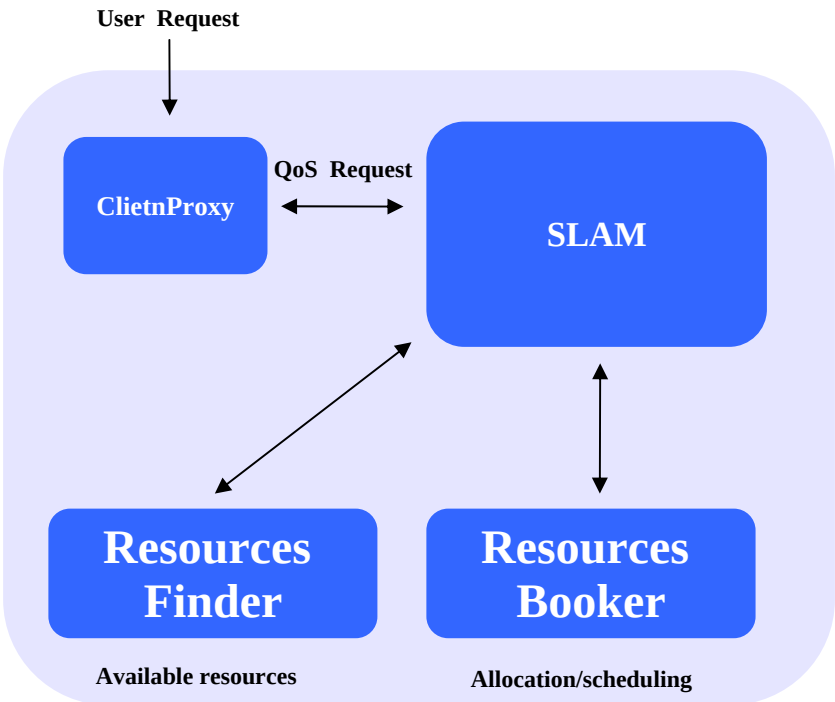
SLA management in Grids

A **Software Level Agreement** is a document that allows to establish the rules for the proper execution of service between the provider and consumer

An SLA manager has to:

1. collect and parse the user requests in order to translate their QoS parameters in term of specific resources requirements
2. discover and choose the best suitable resources according to the scheduling policy adopted.
4. reserve the resources to guarantee that one or more of them will belong only to a user for a given range of time.
5. monitor the service execution to check the QoS requirements fulfillment.

SLA Management Pattern



ClientProxy
Translates the QoS user requests in precise resources requests

SLAM
Checks the availability of resources, reserves the resources, creates and monitors the SLAs

Resources Finder
Interacts with the information sources

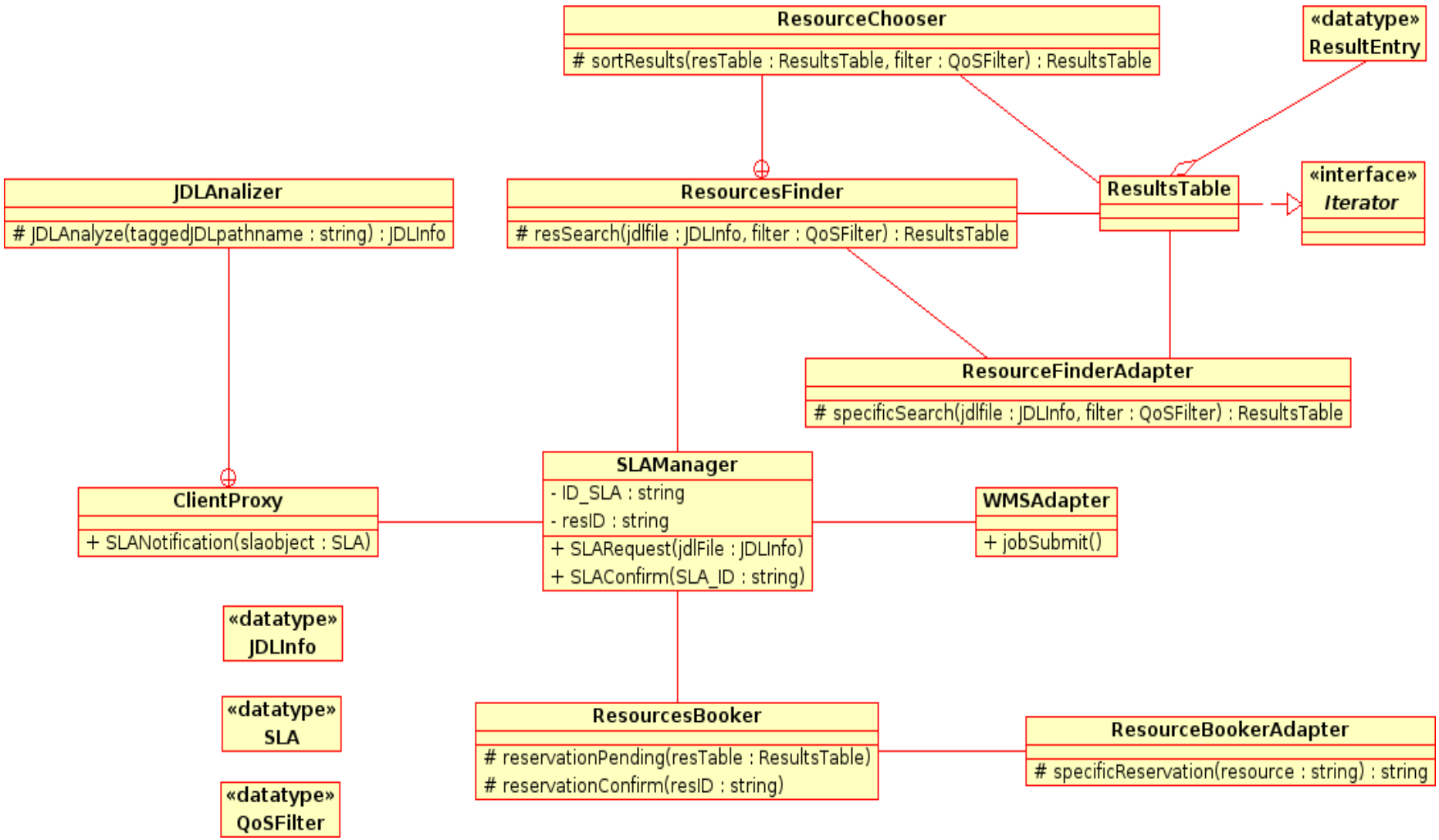
Resources Booker
Interact with the reservation manager

SLAM on gLite

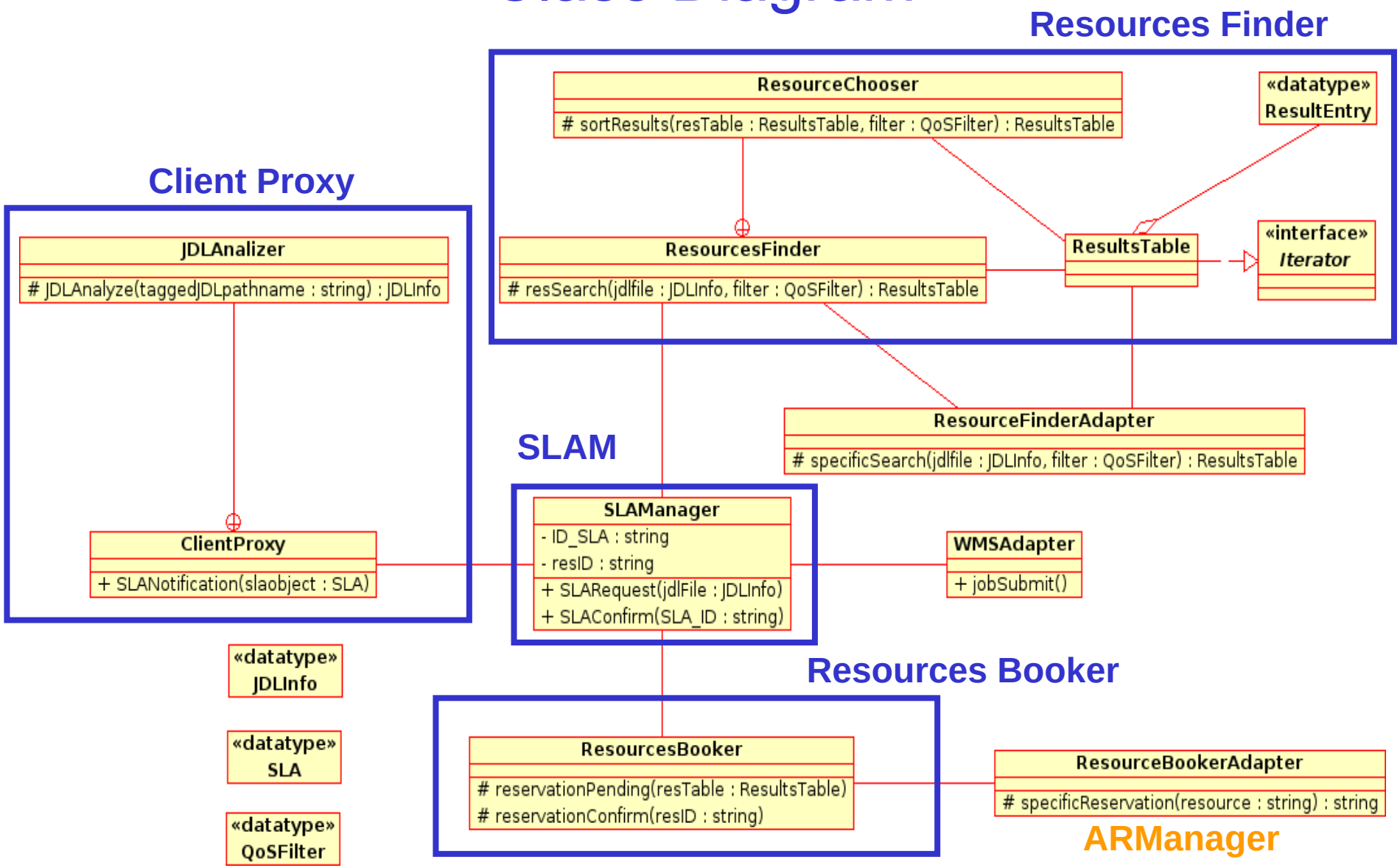
The Service Level Agreement Manager:

- provides to the users the ability to request a service execution under specific constrains
- provides a common interface to query different underlying information services.
- searches among the available resources the most suitable for a specific request (using complex management policy)
- provide a common interface to interact with different underlying CE schedulers

Class Diagram



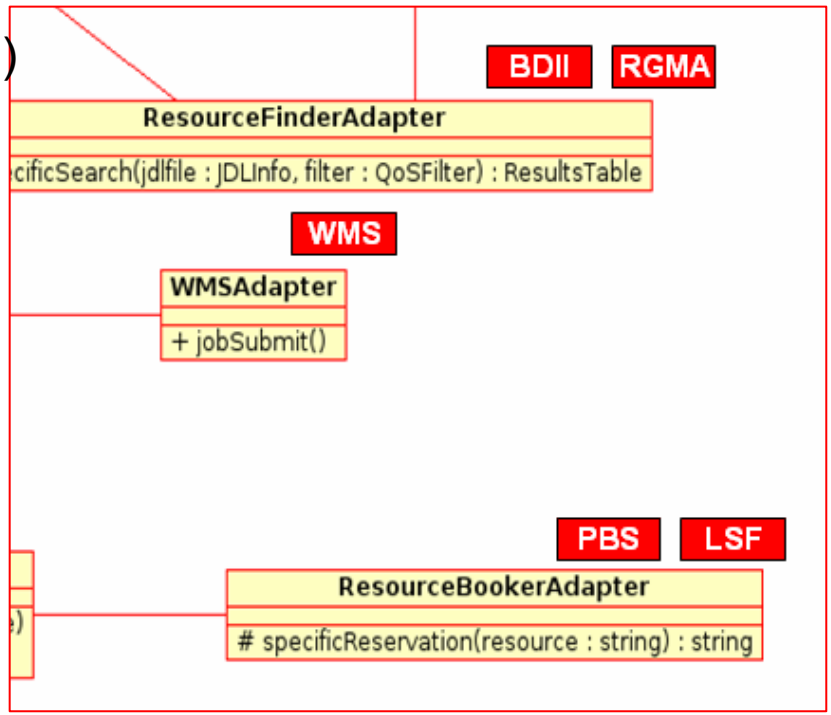
Class Diagram



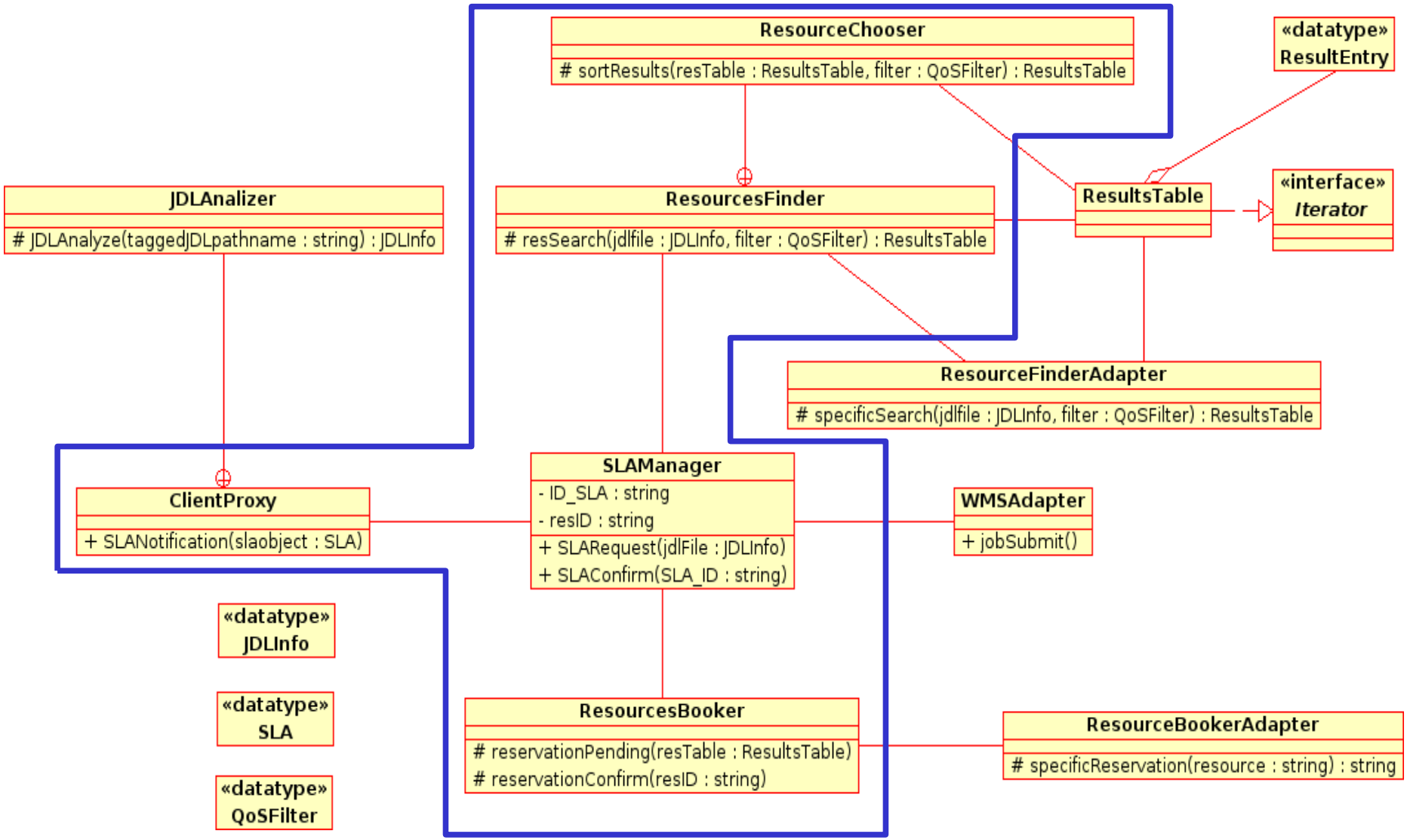
Middleware Dependent Interaction

The framework has to interact with a limited number of **specific middleware components (Adapter).**

- Through **APIs** (BDII, RGMA, PBS, LSF, WMS)
- Sending the input using **command line**



Class Diagramm



Web Services layer

Conclusion

The adopted **Reservation** mechanism allows**:**

- Best effort service : the standard fair share policy for resource management
- **Guaranteed service**: to reserve in a deterministic way specific resources for specific task.
- **Advanced Reservation**

The **framework core** is:

- platform independent
- Java Web Service based

This make it **usable** on **each grid middleware**.

Only few components are middleware dependent.

Conclusion

Users diversification

Each user can **negotiate autonomously** its **SLA**.

He is **free to the constraints** imposed by the *a priori* agreements between Virtual Organizations and Site Administrators.

Jobs diversification

Each user can negotiate **an SLA for each job** (or jobs group), thus creating **different policies** for the management of its jobs.

Software?

Done:

- SLA Manager
- Resources Finder
- Resources Booker
- ClientProxy

High Level

- Planning Table
- Table Manager

Low Level

Under tests:

- ARManager
- JAM

Works in progress:

- Adapter BDII, LSF, WMS using API

Open Grid Forum

Catania March 02, 2009

QoS management in Grids

Di Stefano A., Morana G., Zito D.

Dep. of Computer science and Telecommunication Engineering

Engineering Faculty - Catania University

Thank you for your kind attention

Catania University

Cometa Consortium