Design and Implementation of a Local Scheduling System with Advance Reservation for Co-allocation on the Grid

National Institute of Advanced Industrial Science and Technology (AIST)

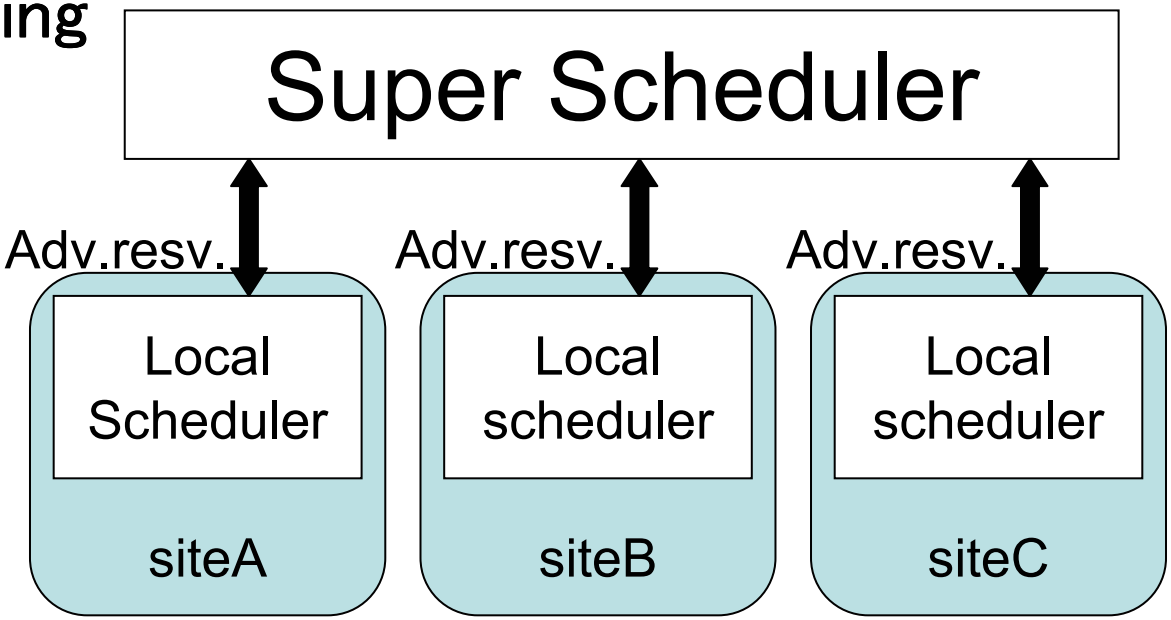
Hidemoto Nakada, Atsuko Takefusa, Katsuhiko Ookubo,

Makoto Kishimoto, Tomohiro Kudoh, Yoshio Tanaka,

Satoshi Sekiguchi

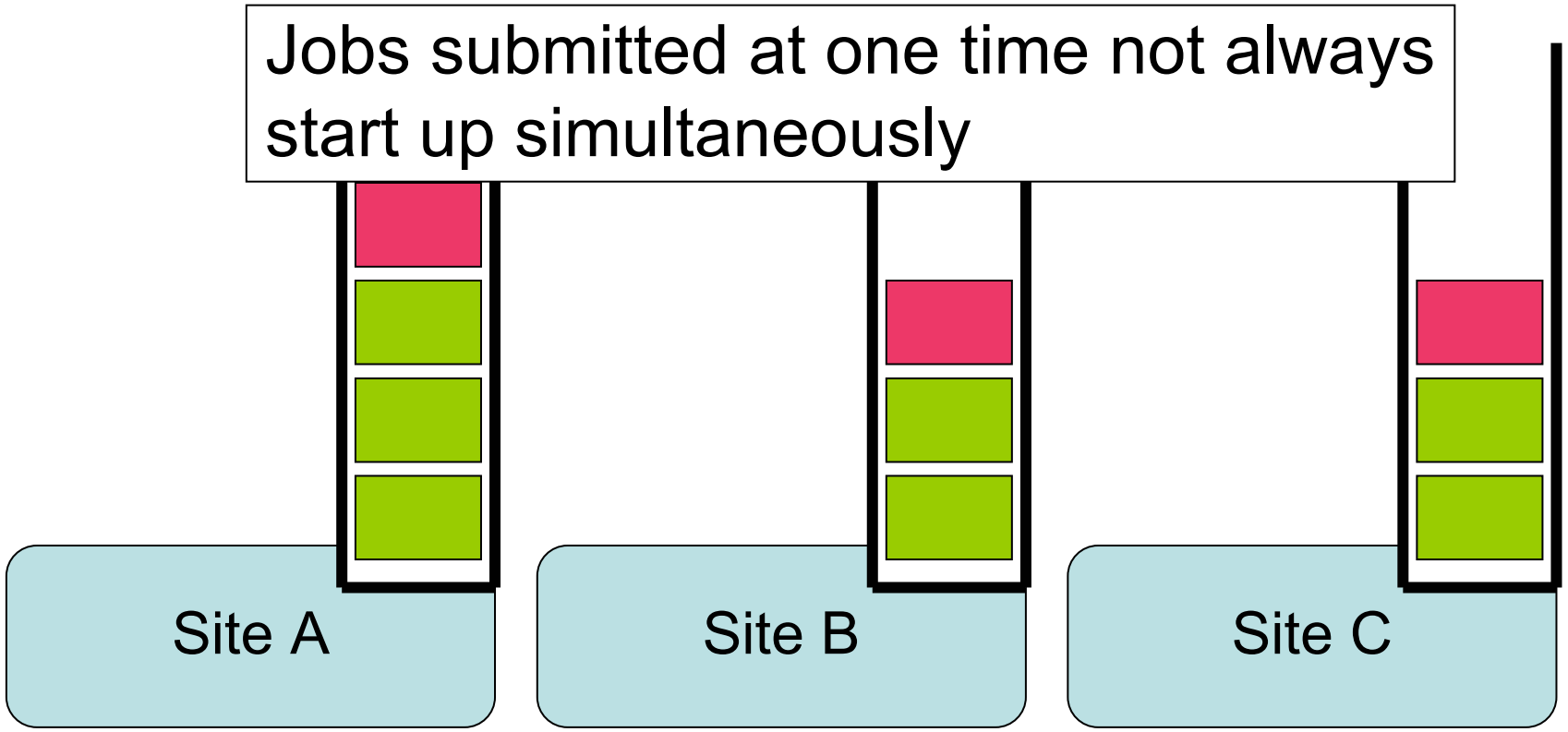
Background

- Large scale computing with resources on the Grid
  - Requires resource co-allocation
- Most sites deploy local queuing system with FCFS (First Comes First Served) + Priority based scheduling



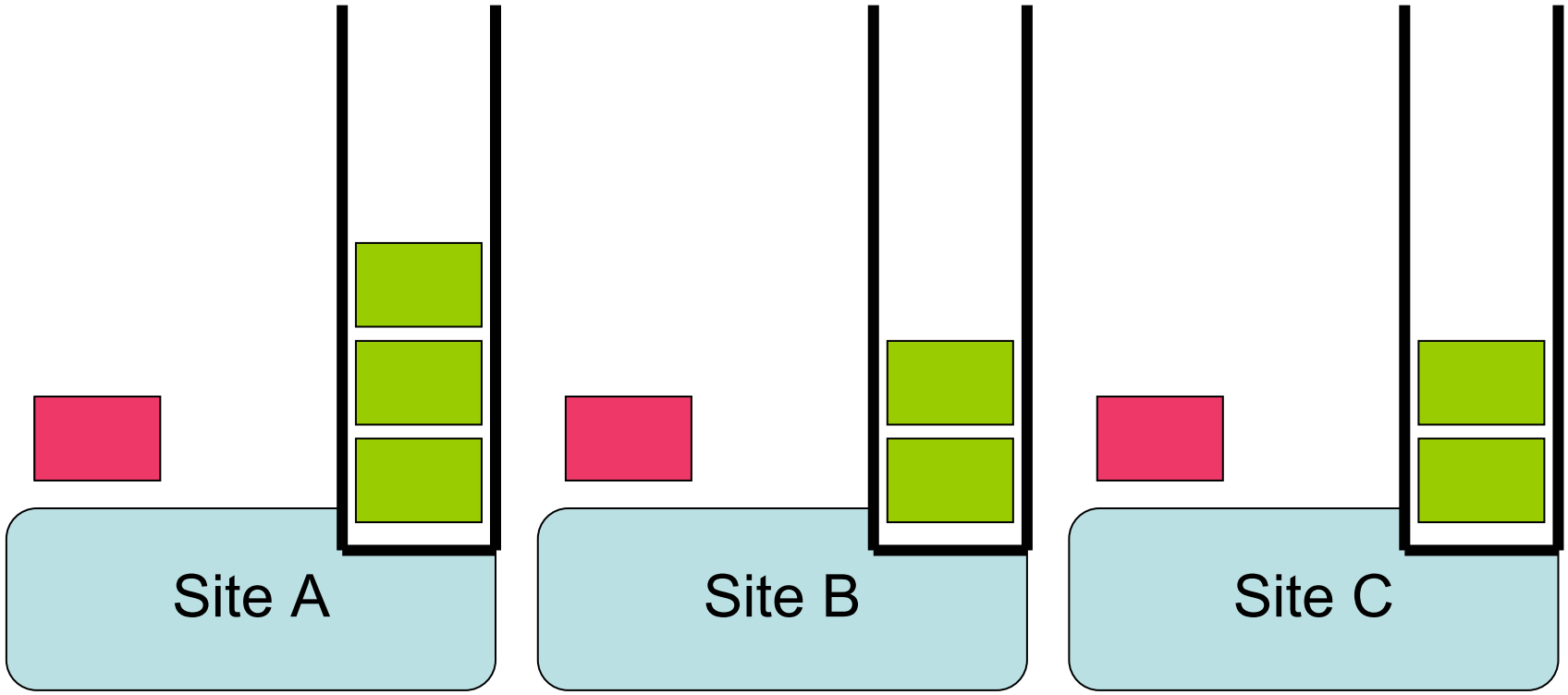
Why do we need Advance Reservation (1/2)

- FCFS
  - Start jobs in the order they are submitted



Why do we need Advance Reservation (2/2)

- Advance Reservation
  - Provide reserved time-slot independent of the FCFS based queue

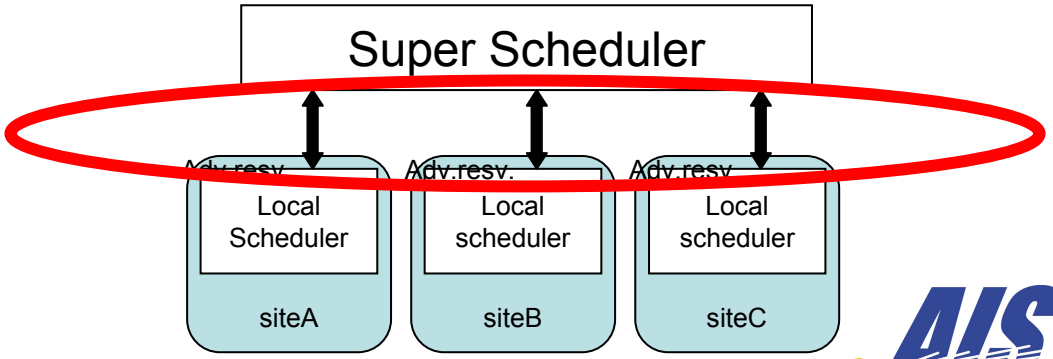


Existing Local Schedulers with Advance Reservation

- Commercial schedulers
  - PBS Professional, LSF
  - Expensive
  - We cannot tweak with the scheduling policy
- Free Scheduler
  - Maui Scheduler
    - Add on scheduler for TORQUE
    - Widely used in the community
    - Source can be modified, but
      - it will be really hard because no API is provided

The Goal

- Implement local scheduler with advance reservation
  - as an add-on module for TORQUE
- Provide external interface for coordination with Super Schedulers
  - WSRF based reservation protocol
    - Globus Toolkit 4 (GT4) authentication
  - Coordination with GRAM (job submission interface)

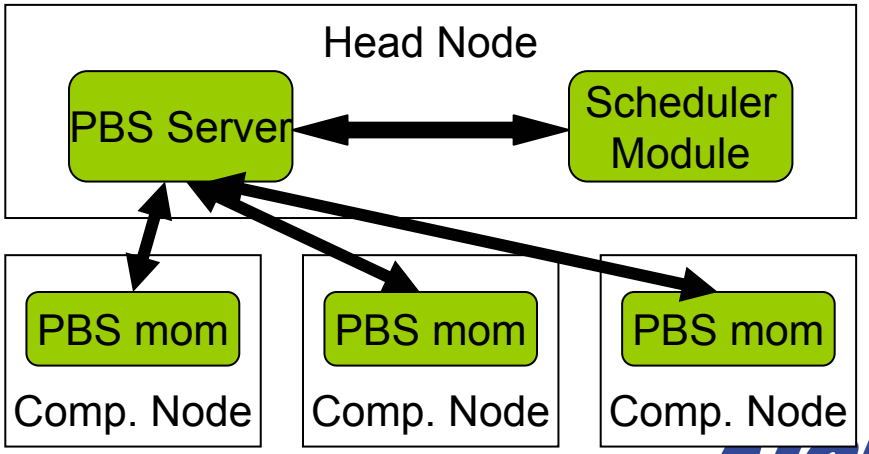


Overview of the talk

- TORQUE
- Proposal of a system
- External Interface for reservation
- Measurement
- Conclusion and Current Status

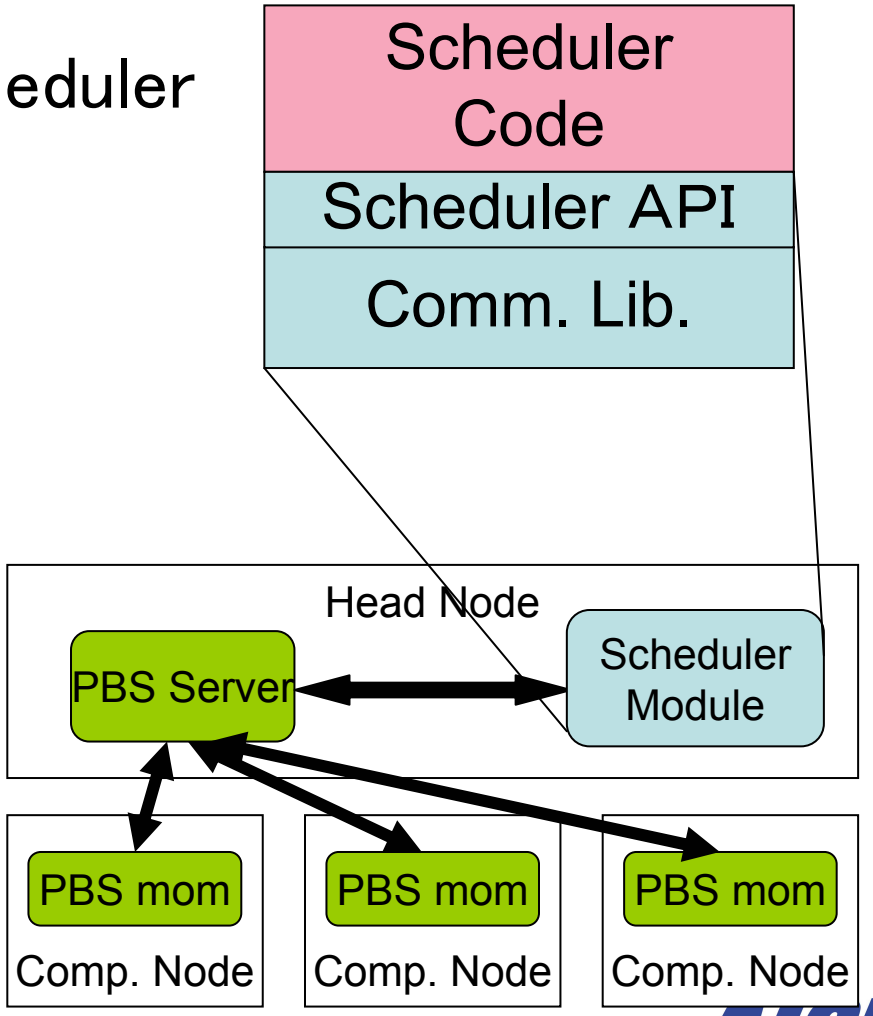
TORQUE

- An OpenPBS descendant
  - c.f. OpenPBS: not maintained any more
  - Can be modified and redistributed freely
- Consists of 3 types of daemons
  - PBS Server
    - Central Server / one for each pool
    - Manages queue and Compute nodes
  - Scheduler module
    - One for each pool
    - Responsible for allocation of job for each nodes.
    - Works upon requests from the PBS Server
  - PBS Mom
    - On Compute Nodes
    - Job invocation, monitoring



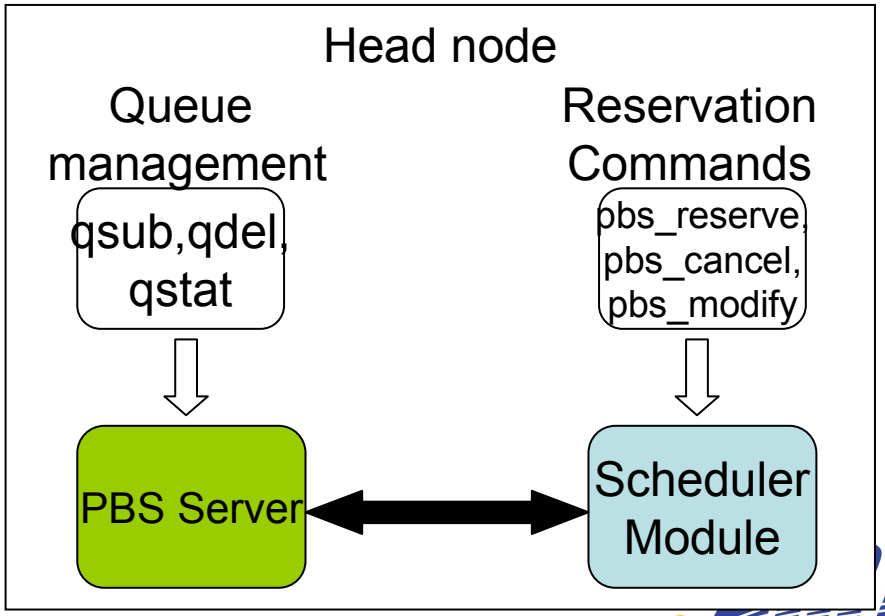
Proposed Architecture

- Implement scheduler
  - Replace the original scheduler
  - written in Java
    - Provides API
- Comm. with PBS Server
  - Protocol
    - Simple text-based
  - Authentication
    - Privileged port base



Scheduler Module Implementation

- **Reservation table is implemented in the Scheduler module**
- **Provides command line interface for reservation**
  - Talks with the Scheduler module
  - Via Java RMI
- **Table serialization**
  - so that reservations can survive reboot of the Scheduler Module
  - With db4objects



Command line interface

- pbs_reserve
  - Requests reservation
  - Input : start, end, num. of nodes
  - Output : reservation ID
- pbs_rsvcancel
  - Canceel reservation
  - Input: reservationID
- pbs_rsvstatus
  - Printout status of reservation
  - Input : reservationI
  - Output : reservations status
- pbs_rsvmodify
  - Modify reservation
  - Input : reservationID, Start, end, num. of nodes

Usage scenario

- Make reservation

```
> pbs_reserve -s 12:00 -e 14:00 -n 1
Reserve succeeded: reservation id is 14
```

- Check the status

```
> pbs_rsvstatus
 id owner          start             end     duration state
 14 nakada     Feb 20 12:00  Feb 20 14:00     2h00m Confirmed
```

- Submit a job with ReservationID

```
> qsub -W x=rsvid:14 script
```

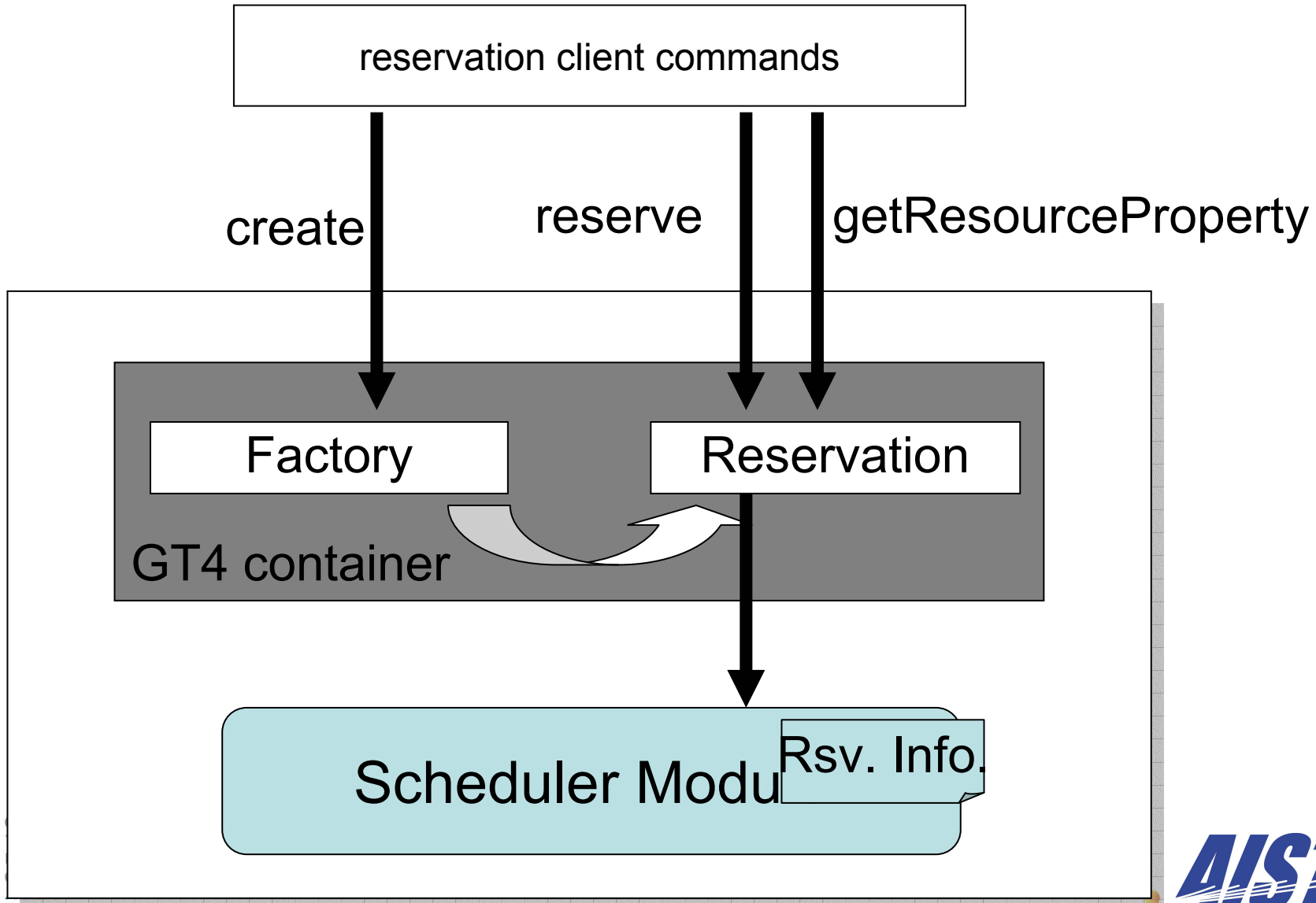
External interface for the reservation

- To enable co-allocation of multiple resources, reservation capability have to have external interface
  - Security
  - Standardization in the future
- Employ GT4 and WSRF
  - Security
    - Authentication with PKI
    - Authorization with the Grid Map File
  - WSRF (Web Services Resource Framework)
    - Standardization with OASIS
    - interface will be specified with WSDL

WSRF based Interface

- Factory Service
  - Creates Reservation Services
  - **CreatePBSReservation**
    - Input – reservation timeframe, # of nodes
    - Output – EPR (pointer) to the created
    - Reservation Service
- Reservation Service
  - have the reservation status as the resource property
  - **reserve** : Make reservation
  - **cancel**: Cancel the reservation
  - **modify**: Modify the reservation
  - **getStatus**: Update the info. in the resource property
  - **getResourceProperty**:
    - retrieve the status info from the property

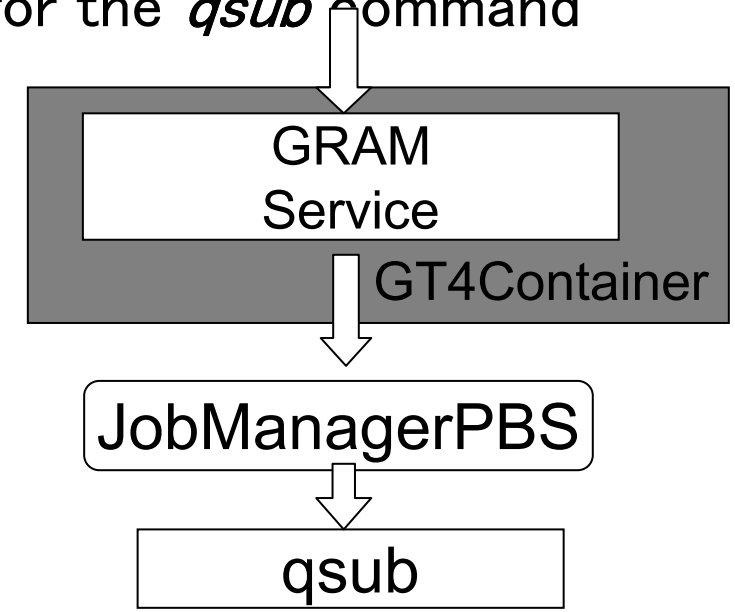
WSRF based reservation service



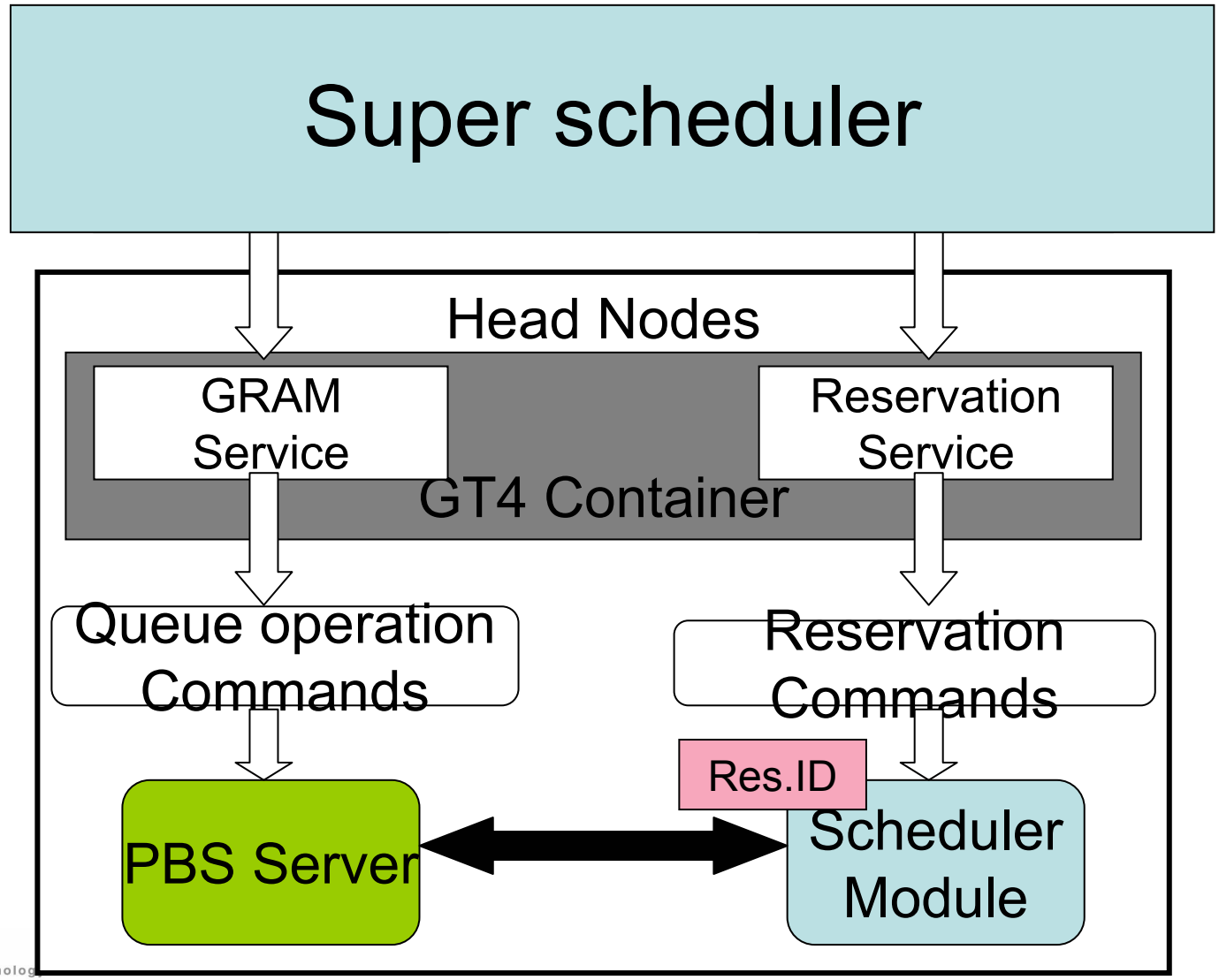
Coordination with GRAM

- Specify the Rsv. ID via GRAM
  - Embed the Rsv.ID in the Job description
    - GRAM have extension syntax
    - (slightly) Modified GRAM PBS Job Manager
    - so that it gives the Rsv.ID for the *qsub* command

```
<extensions>
  <schedulerAttrs name="reservationID">
       xxxxxxxxxx
  </schedulerAttrs>
</extensions>
```



The Big Picture



Measurement

- Measured time spent to make and cancel reservation
  - Direct access, via GT4
- Environment for the measurement
  - All modules are running in a single node
    - PBS server, GT4 Container, Client
  - Pentium III 1.4 GHz, 2CPU, 2Gbyte

	reserve	cancel
Direct Access	0.78 s	0.68 s
Via GT4	1.7 s	1.3 s

- Direct Access cost
  - RMI library loading cost on the client side
- GT4 overhead
  - authentication / authorization

Conclusion

- Designed and implemented local scheduler capable of advance reservation
  - Based on TORQUE
- External interface implemented on Globus Toolkit 4
  - Expose the reservation capability
  - Coordination with GRAM service

Current Status

- SGE implementation has been done.
  - works completely outside of the SGE
  - does not replace scheduler module, leveraging queue management interface
- Updated WSRF interface
  - now it allows 2-phased protocol for safe transaction
- Will be available shortly from

http://www.g-lambda.net/plus

Acknowlegement

This work is partly funded by the Science and Technology Promotion Program's "Optical Paths Network Provisioning based on Grid Technologies" of MEXT, Japan.