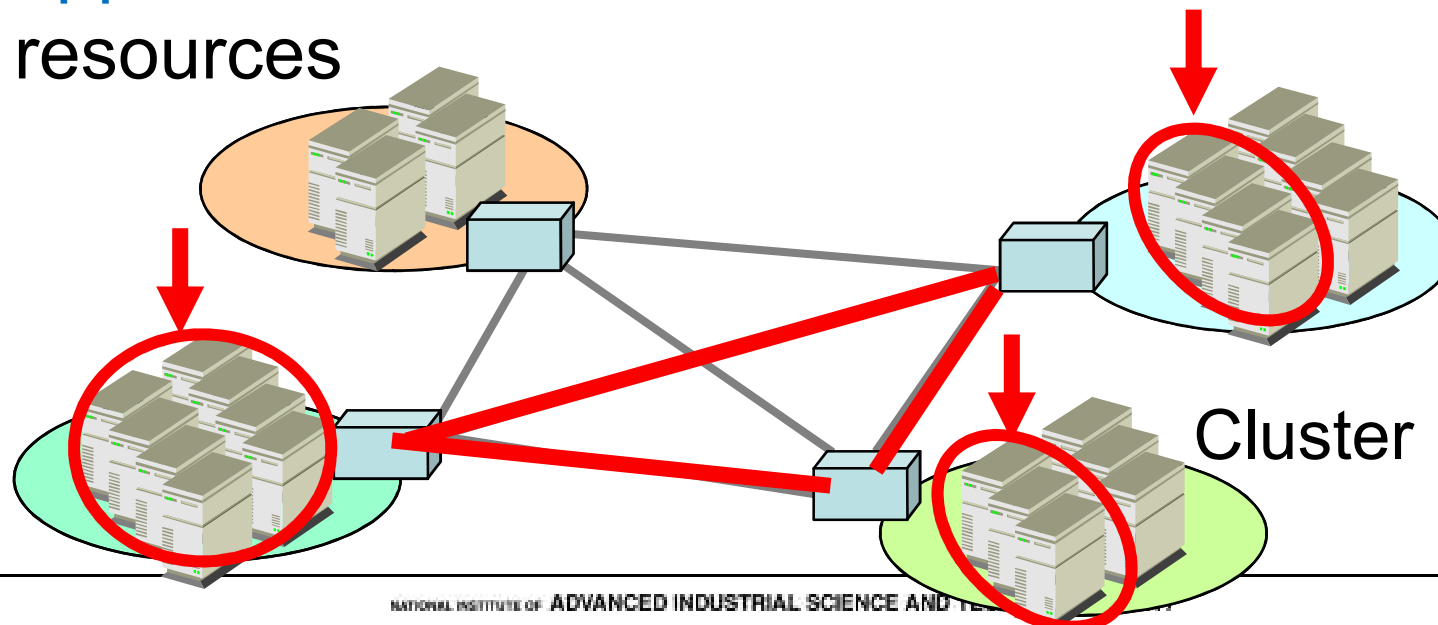


GridARS: An Advance Reservation-based Grid Co-allocation Framework for Distributed Computing and Network Resources

Atsuko Takefusa, Hidemoto Nakada, Tomohiro
Kudoh, Yoshio Tanaka, and Satoshi Sekiguchi
National Institute of Advance Industrial Science and Technology (AIST)

Metacomputing over Computational Grids

- Grid technologies allow large-scale parallel computing over distributed resources managed by different organizations
- A crucial issue for achieving high effective performance of **file-grain message passing applications** is **Grid co-allocation** of various resources



Current Strategies of Grid Co-allocation

- (1) Manual reservation and job execution by SSH
 - Difficult to use resources effectively
 - Unrealistic to expect to have a local account on all of the available resources on Grids
- (2) Manual reservation of resources managed by resource schedulers
 - Administrators configure a reservation queue
 - Allows resource management based on site policies
 - Many manual configuration errors reported
- (3) Automatic reservation of resources managed by resource schedulers
 - Allows individual resource management, as well as (2)
 - Avoid human configuration errors

Issues for Grid Co-allocation (1/2)

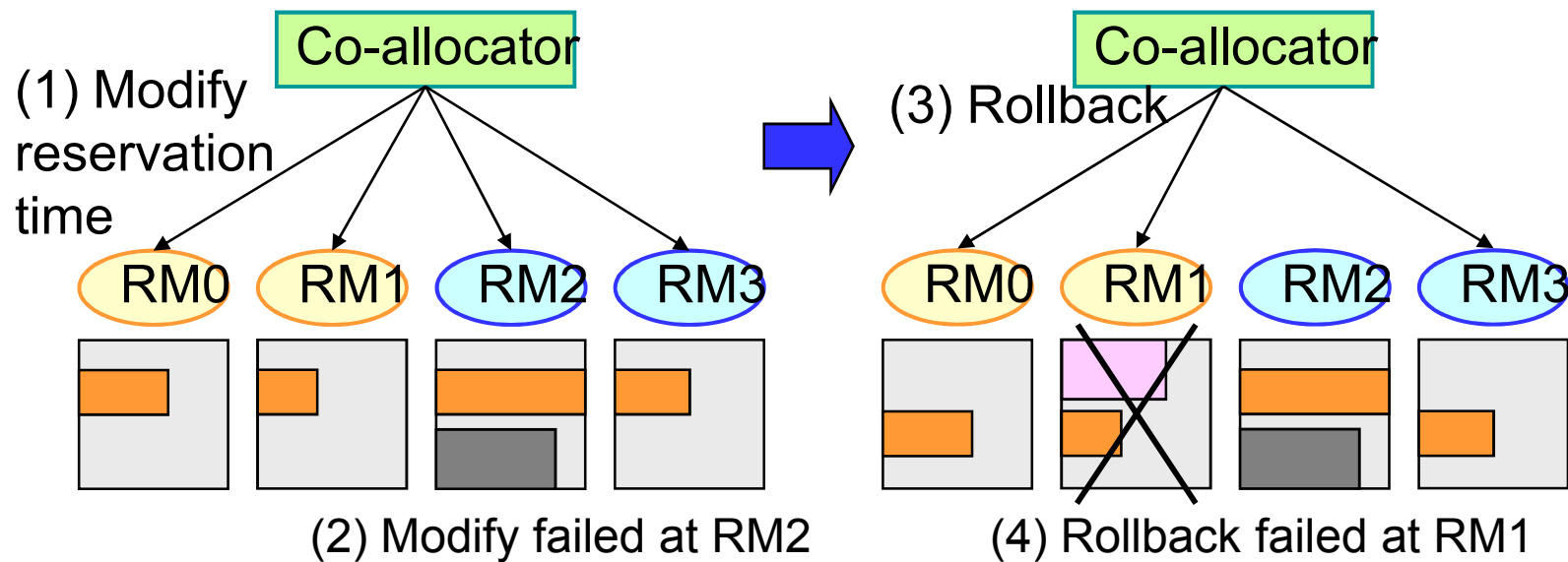
- **Co-allocation of various resources**
 - HPC parallel applications on Grids require not only computing resources, but also network resources, such as bandwidth
- **Coordination with existing resource schedulers**
 - Grid resources have been managed by existing resource schedulers, such as GridEngine, TORQUE, or commercial batch queuing systems
- **Advance reservation**
 - Difficult to estimate when a user job will start in each queuing system.
 - An advance reservation capability is required

Issues for Grid Co-allocation (2/2)

- Standard I/F
 - Existing resource schedulers provide various I/F, such as CLI or GUI
 - Schedulers should provide a standard I/F with secure communication for various global users
- Two-phase commit (2PC)
 - Resource schedulers should support **2PC reservation I/F** so that global schedulers can **allocate distributed resources simultaneously** based on transactions

Why do we need 2-phase commit?

- One-phase commit might cause failure on modification of a existing reservation



Contribution

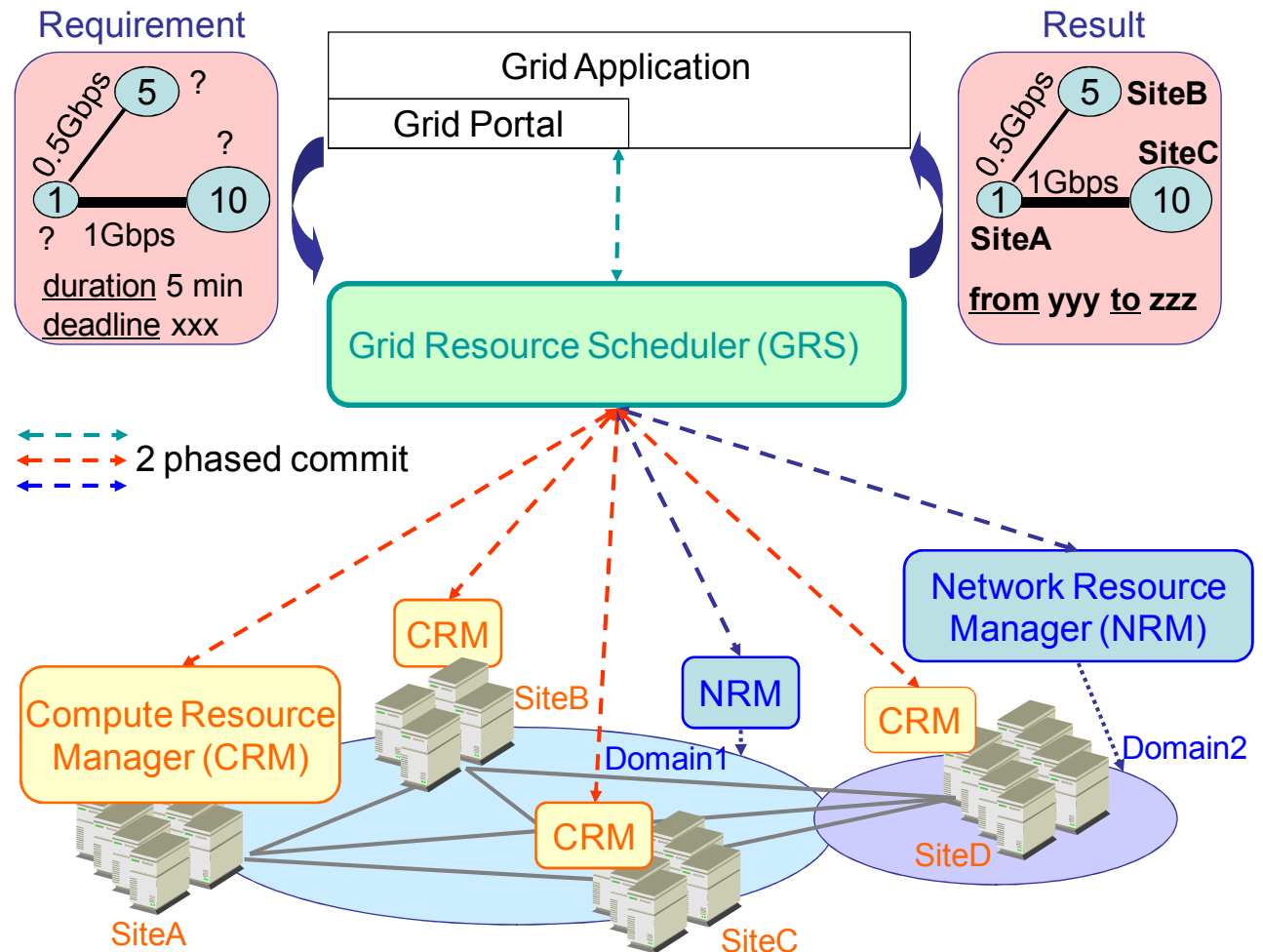
- Propose GridARS
(Grid Advance Reservation-based System Framework)
 - **Grid co-allocation framework** for distributed resources, such as computers and network
 - **General 2PC advance reservation protocol over WSRF** (Web Service Resource Framework) for global coordinators and existing resource schedulers
 - GridARS-Coscheduler
 - Finds suitable resources and co-allocates them by distributed transaction
 - GridARS-WSRF
 - Interface module for the proposed two-phase commit (2PC) reservation protocol
 - Reference implementation called GridARS-WSRF/GT4 has been developed using Globus Toolkit 4 (GT4)

The Rest of the Talk

- Architecture of GridARS Grid co-allocation framework
- Performance of GridARS 2PC reservation process
- Case study: demonstration using GridARS
- Related work
- Conclusions and future work

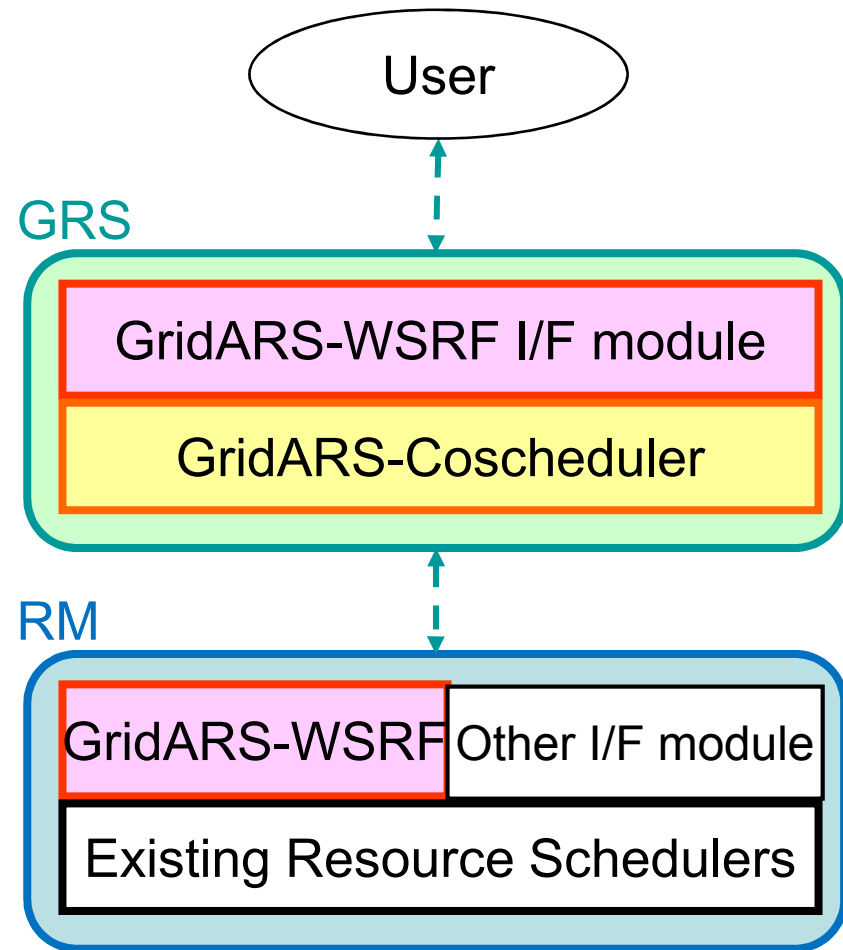
GridARS Grid Co-allocation Framework

- Consists of GRS and RMs (CRM, NRM)
- In each RM, existing schedulers manage a reservation table
- GRS co-allocates suitable resources in coordination with RMs
- Provides hierarchical 2PC
→GRS can be one of RMs



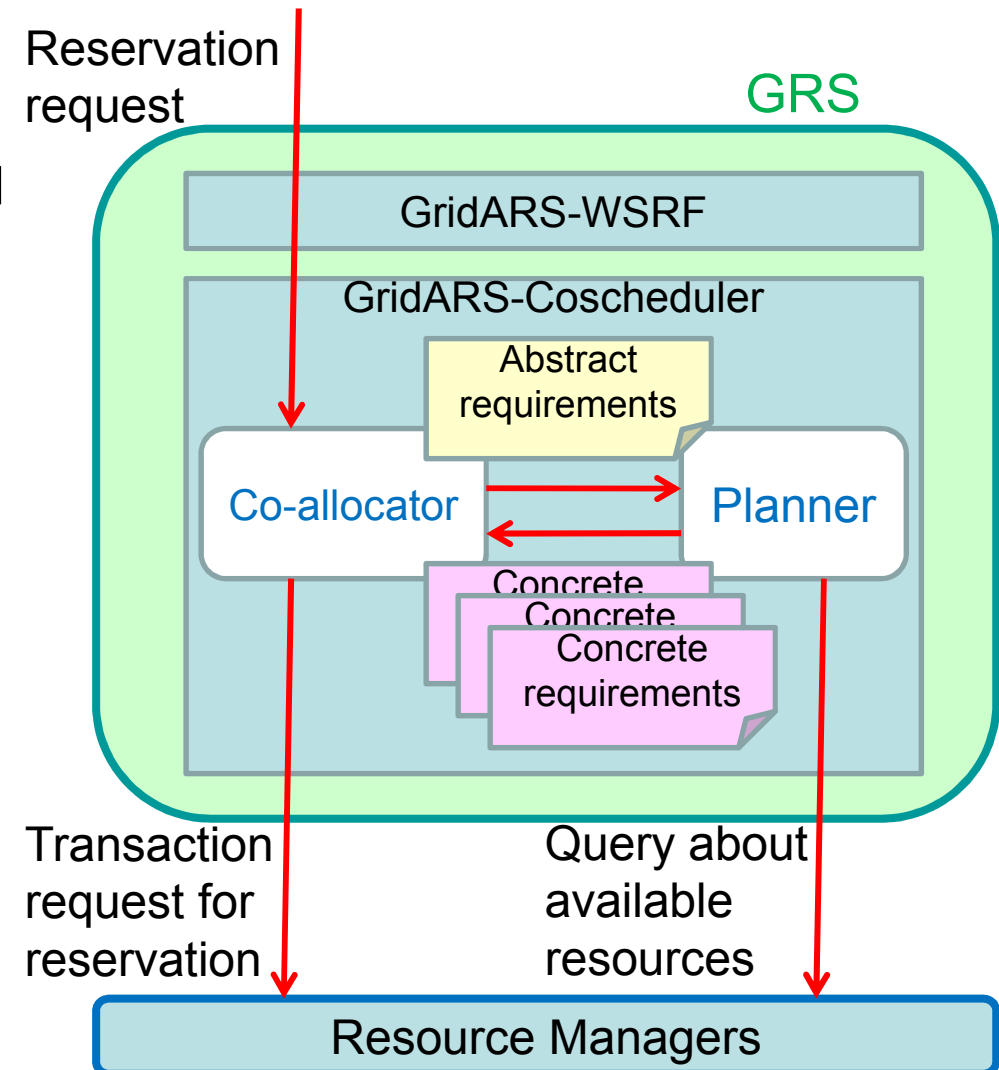
GridARS System Architecture

- GRS consists of **GridARS-Coscheduler** and **GridARS-WSRF**
- RM consists of **GridARS-WSRF** and **existing resource schedulers** with advance reservation



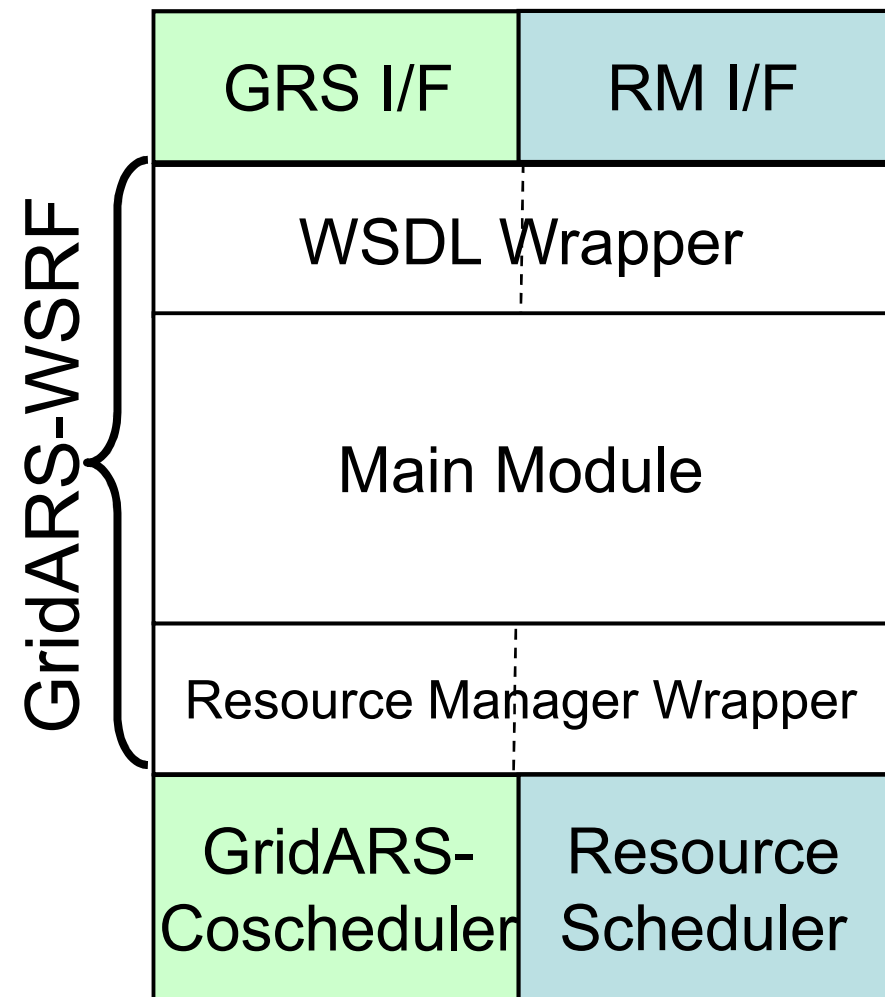
GridARS-Coscheduler

- **Co-allocator**
 - Receives user request and sends it to Planner
 - Negotiates with RMs and books resources selected by Planner simultaneously
- **Planner**
 - Determines candidates from among concrete resources
 - Returns the result to Co-allocator
 - Replaceable for various strategies



GridARS-WSRF

- Provides a polling-based 2PC I/F for AR
 - Enables asymmetric communication
 - Applies Notification, optionally
- Consists of **WSDL Wrapper**, **Main Module**, and **RM Wrapper**
 - **WSDL Wrapper** is in between various resource parameters (e.g. JSDL) and **Main Module**
 - **Main Module** enables a polling-based 2PC AR process in a non-blocking manner
 - **RM Wrapper** provides API



Design of GridARS-WSRF

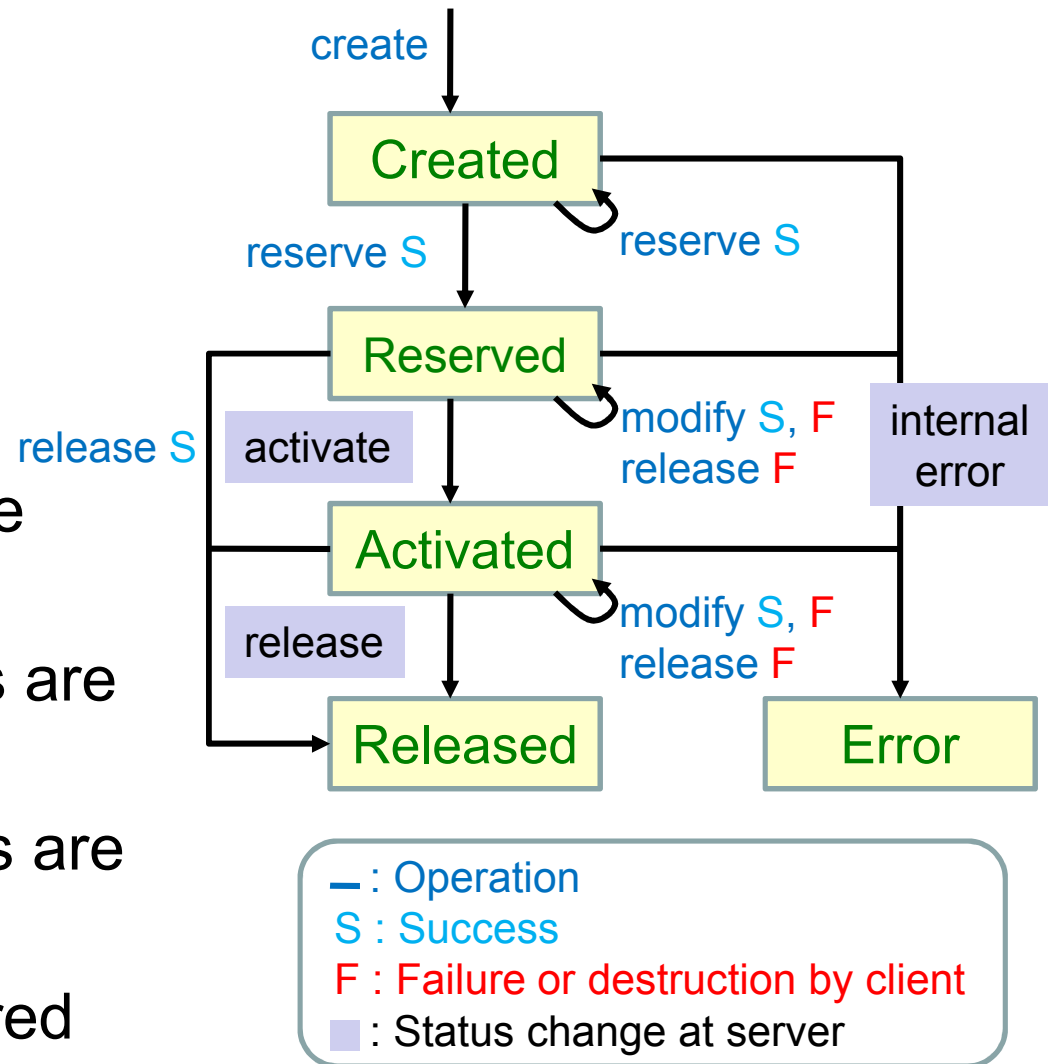
- GridARS-WSRF Provides:
 - [ReservationFactoryService](#) for service reception and information service
 - [ReservationService](#) for AR operations and the resource management
 - [ReservationCommandService](#) for 2PC
- Based on [GNS-WSI2](#)
 - Grid Network Service - Web Services Interface
 - Defined by the [G-lambda](#) project
 - AIST, KDDI R&D Laboratories, NTT, and NICT
 - Web Services-based I/F for network resources for Grid middleware and applications

Service Operations related to Reservation, Modification, and Release

- ReservationFactoryService
 - `create` : Creates ReservationResource (RR), a service instance for each reservation and returns the EPR
 - `getAvailableResource` : Returns available resource info
- ReservationService
 - `reserve / modify / release` : Receives reserve/modify/ release request, creates ReservationCommandResource (RCR), and returns the EPR
 - `getReservationStatus` : Returns reserved resource status
 - `getResourceProperty(GridResource)` : Returns the reservation result
- ReservationCommandService
 - `commit / abort` : Completes / destroys the reservation process
 - `getReservationCommandStatus` : Returns current operation status

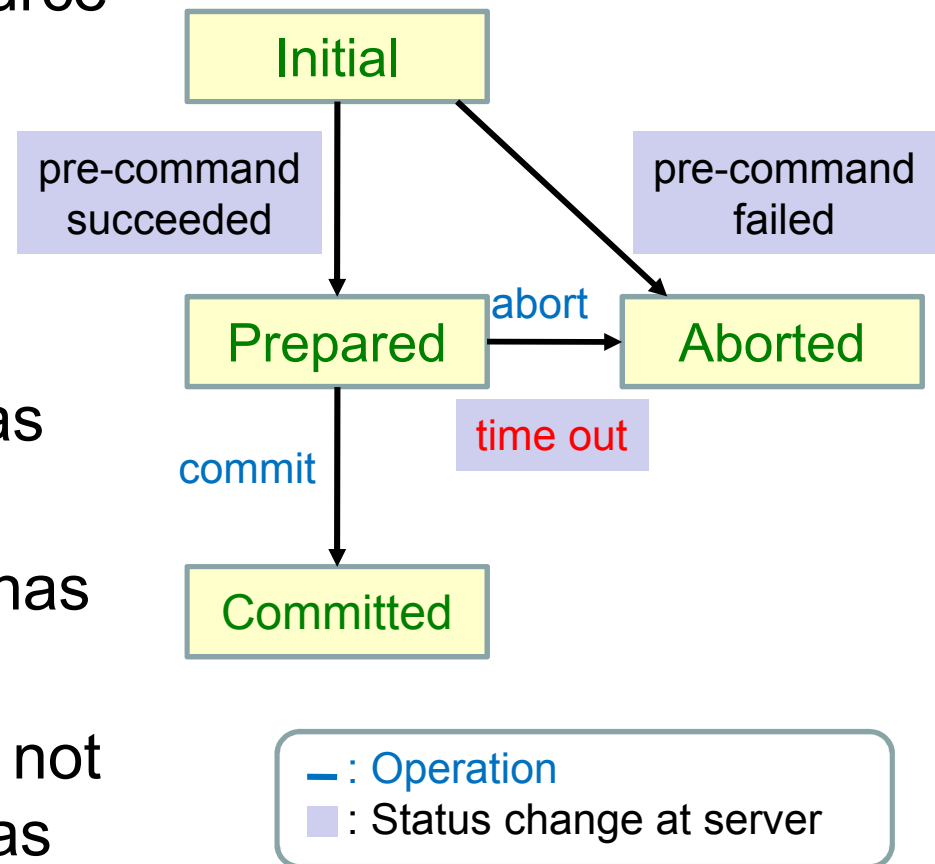
ReservationStatus Transition

- Property of ReservationResource
- Represents the current reservation status
- **Created** : RR is created
- **Reserved** : Resources are booked
- **Activated** : The resources are activated
- **Released** : The resources are released
- **Error** : Errors have occurred

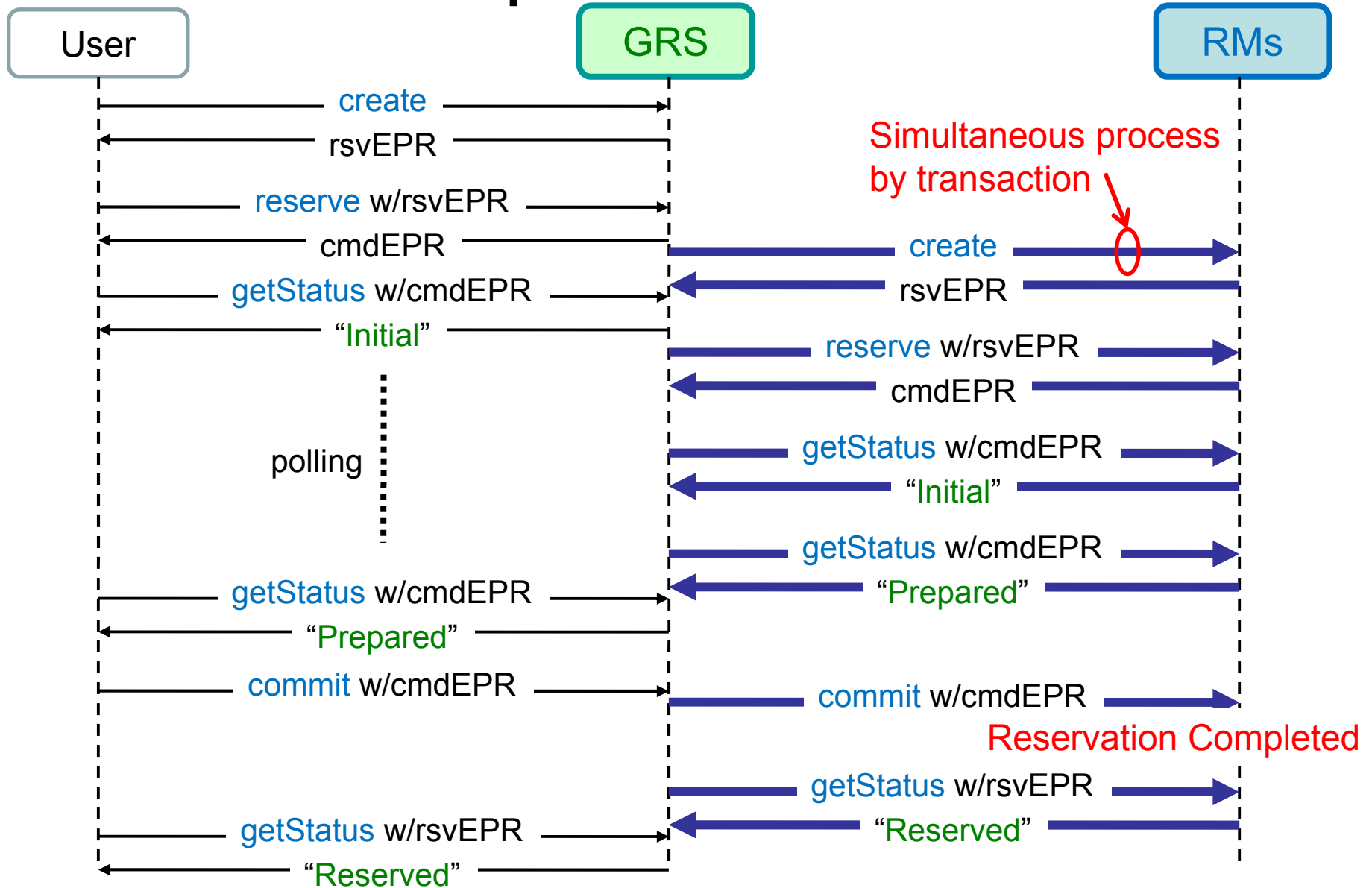


ReservationCommandStatus Transition

- Property of ReservationCommandResource
- Represents the current command status
- **Initial** : Command has been received, but not completed
- **Prepared** : The command has been prepared
- **Committed** : The command has been completed
- **Aborted** : The resources are not available or the command has expired (applied “**time out**”)



Protocol Sequence of AR Process



GridARS-WSRF/GT4

- An implementation of GridARS-WSRF
 - Using **Globus Toolkit 4 (GT4)**
 - **GSI** (Grid Security Infrastructure)
 - Authentication based on PKI
 - Authorization with “grid-mapfile”
 - GRS adopts GSI delegation capability
- Use existing resource descriptions
 - **JSDL** for computing resources
 - Resource description of **GNS-WSI2** (by G-lambda) for network resources
 - Extend them to represent AR requirements

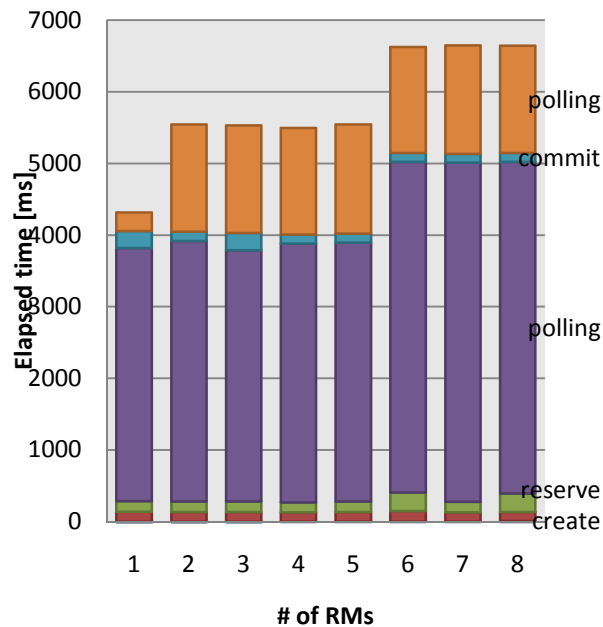
Performance of GridARS 2PC Reservation Process

Performance of GridARS 2PC Reservation Process

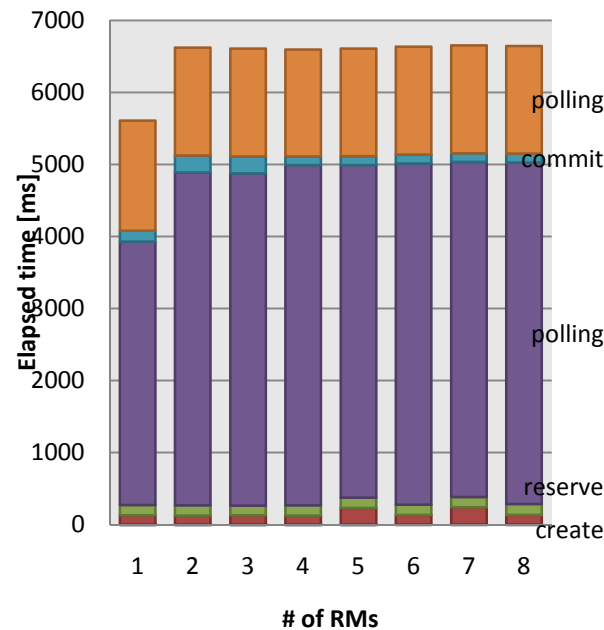
- Present the basic performance to confirm effectiveness of GridARS 2PC over **WSRF/GSI**
- Experiments using GRS and 1-8 RMs
- Comparison of
 - Case1.** Non additional latency between GRS and RMs
 - Case2.** Additional latency on the path to the **1st RM**
 - Case3.** Additional latency on the paths to **all RMs**
- Environment
 - User is located on the GRS node and eight RM nodes are deployed
 - Latencies between GRS and RMs = 0.2[ms]
Additional latencies = 186[ms] (= RTT between Tokyo and NC)
 - 2[sec] for each pre-reservation and 1[sec] for completion at each RM

Elapsed Time of 2PC Reservation Process

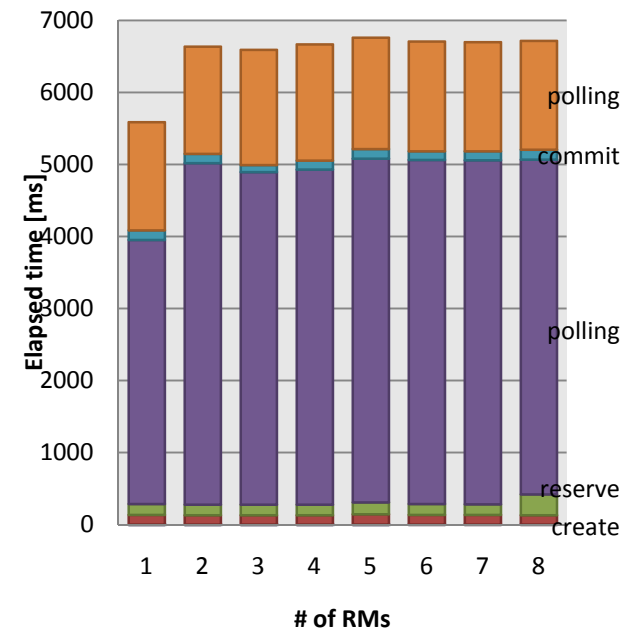
Case1. No additional latency



Case2. Additional latency on the path to RM1



Case3. Additional latencies on the paths to all RMs



- Case2 and Case3 is comparable, and longer than Case1
 - The latest reservation process determines the total elapsed time
- While # of RMs increases, the elapsed times increase because of the load at GRS, but around 6.7[sec] **GridARS 2PC works efficiently on Grids**

Case Study

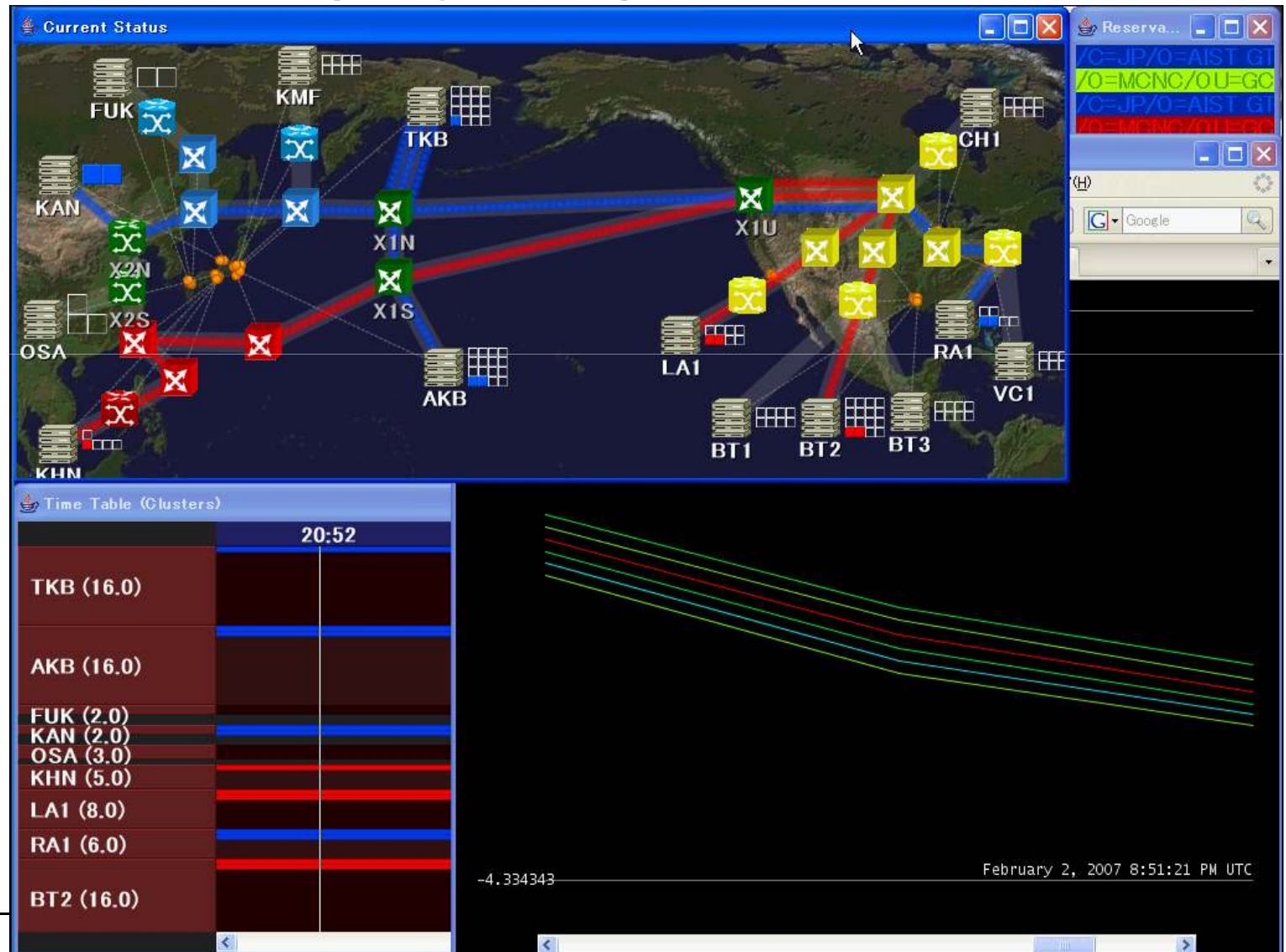
Case Study: a Trans-pacific Experiment using GridARS

- Demonstration at GLIF2006 and SC06 [Submitted to GridNets2007]
 - In cooperation with **G-lambda** and the **EnLIGHTened Computing** project in the US
 - User booked trans-pacific computing and network resources managed by different organizations
 - G-lambda team performed QM/MD simulation developed using GridMPI over trans-pacific resources
- Environment
 - # of sites = 10 (7 in Japan and 3 in the US)
 - # of network domains = 4 (3 in Japan and 1 in the US)
 - CRM : GridARS-WSRF/GT4, PluS + GridEngine (Japan), Maui + TORQUE
 - NRM : NRMs developed by KDDI R&D Labs, NTT, EnLIGHTened, and AIST (EnLIGHTened and AIST used GridARS-WSRF/GT4)
 - US resources were booked via **HARC Co-allocator (EnLIGHTened)**

Trans-pacific Experiment using GridARS

Sent 10[min] reservation request and performed MPI simulation, continuously

GridARS worked stably during the demo



Related Work

- **Moab Grid Suites**
 - Co-allocate computing resources managed by MAUI+TORQUE
 - Commercial and support various tools
- **CSF4**
 - WSRF-based scheduling framework developed using GT4
 - Support AR for commercial LSF clusters
- **GUR**
 - AR-based global scheduler, which works with the Catalina external scheduler
 - Finds and books available resources one by one
- **VIOLA MSS**
 - Will support co-allocation of computing and network resources based on the WS-Agreement standard
- **HARC**
 - Applies the Paxos commit protocol for fault-tolerance
 - REST-styled HTTP messaging

Comparison of Co-allocators

	Target resources	Coordination with existing schedulers	Advance Reservation	WSRF/GSI	2PC
Moab	Comp	✓	✓	-	-
CSF4	Comp	✓	LSF	✓	-
GUR	Comp	✓	✓	- (SSH)	-
VIOLA MSS	Comp, Net	?	✓	(✓)	-
HARC	Comp, Net	✓	✓	- (HTTPS)	✓
GridARS	Comp, Net	✓	✓	✓	✓

Conclusions

- Proposed **GridARS Grid co-allocation framework** for distributed resources, such as computers and network
 - **general 2PC advance reservation protocol over WSRF/GSI** for global coordinators and existing resource schedulers
 - Consists of the GridARS-Coscheduler co-allocator and the GridARS-WSRF I/F module
- The basic performance of GridARS 2PC showed GridARS co-allocation framework and the simultaneous reservation process worked efficiency on the Grid
- The case study showed GridARS could co-allocates trans-pacific computing and network resources stably

Future Work

- Make the reservation protocol more practical
 - SLA, negotiation, WS-Agreement?
- Investigate suitable co-allocation algorithms for multiple resources
- Collaborate with other Grid co-allocation system, such as HARC and VIOLA MSS

Acknowledgements

- The G-lambda project
- The EnLIGHTened Computing project
- This work is partly funded by the Science and Technology Promotion Program's "Optical Paths Network Provisioning based on Grid Technologies" of MEXT, Japan