Global Grid Forum 15, GRAAP Session#3
Boston, October 5, 2005

A MetaScheduling Service for Co-allocating Arbitrary Types of Resources

Oliver Wäldrich, Wolfgang Ziegler, FhG, Philipp Wieder, FzJ

SCAI
Fraunhofer Institut
Algorithmen und Wissenschaftliches Rechnen

Forschungszentrum Jülich
in der Helmholtz-Gemeinschaft

Core GRID

VIOLA

Acknowledgements

Some of the work reported in this presentation is funded by the German Federal Ministry of Education and Research through the VIOLA project under grant #123456. This presentation also includes work carried out jointly within the CoreGRID Network of Excellence funded by the European Commission's IST programme under grant #004265.

SCAI

Fraunhofer Institut
Algorithmen und Wissenschaftliches Rechnen

Forschungszentrum Jülich
in der Helmholtz-Gemeinschaft

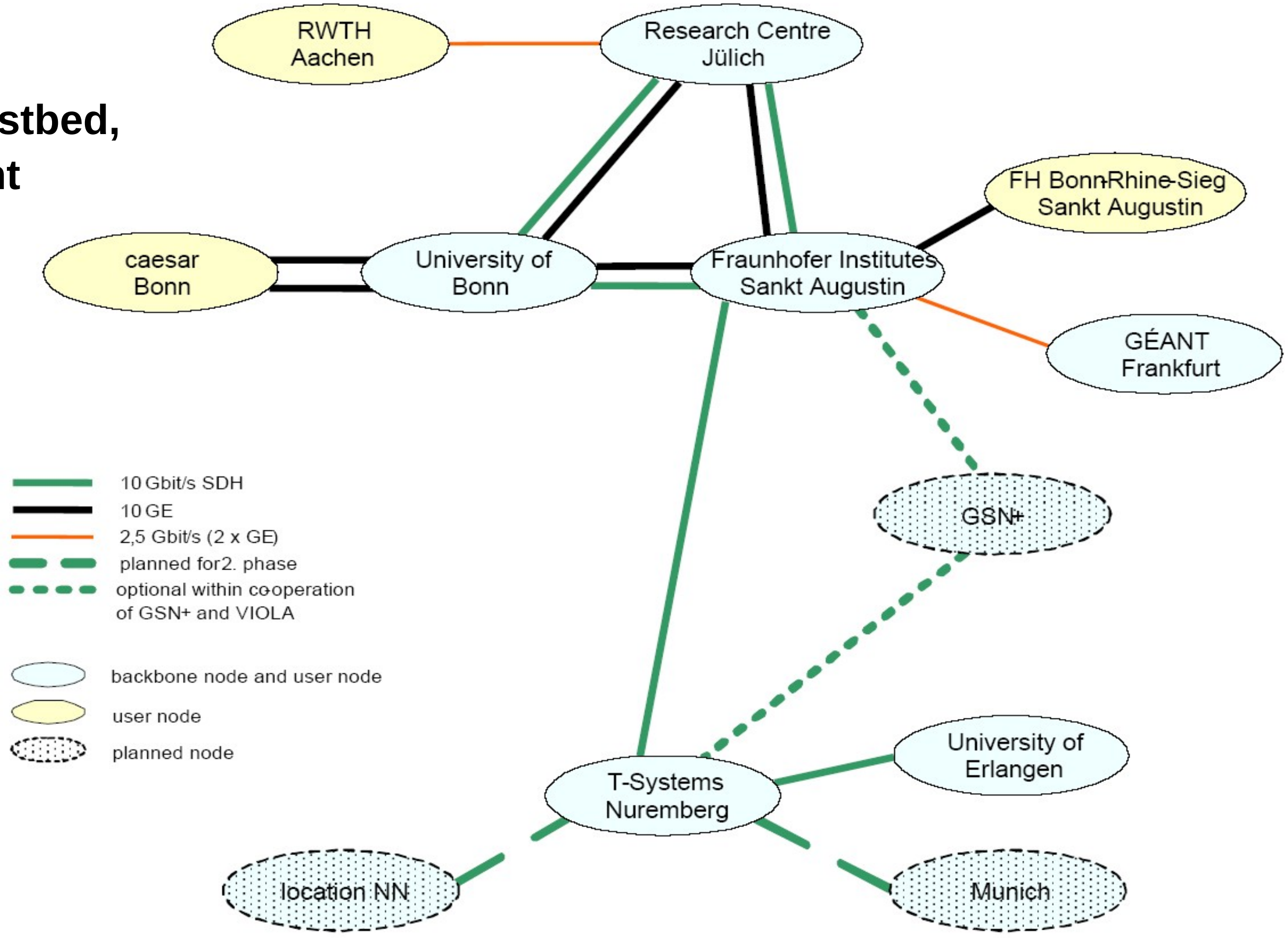
CoreGRID

VIOLA

VIOLA Network

VIOLA-Networking

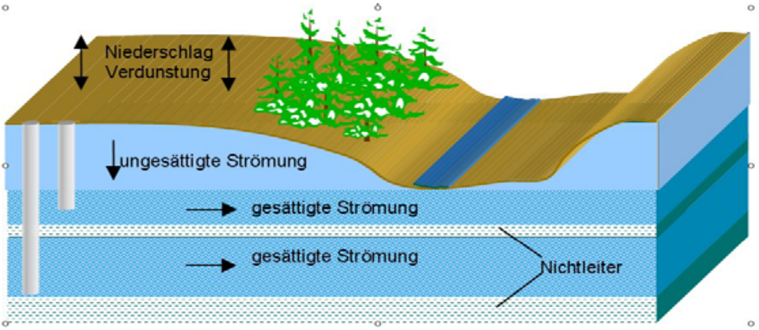
- Deployment and operation of the testbed, test of advanced network equipment
- Signaling and reservation
    - bandwidth- and QoS-reservations in the network
    - interfaces for user-driven reservation: immediate and in advance

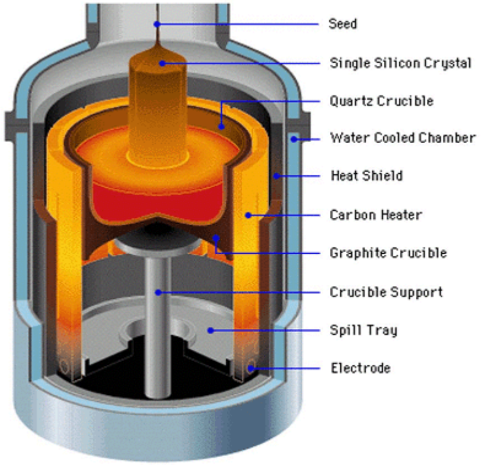


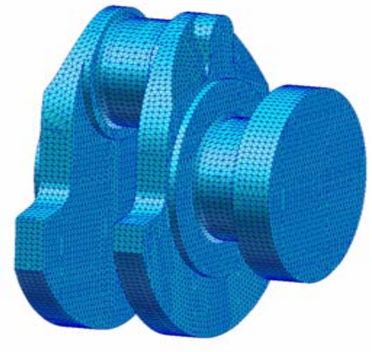
VIOLA Subprojects: Distributed Parallel Simulations

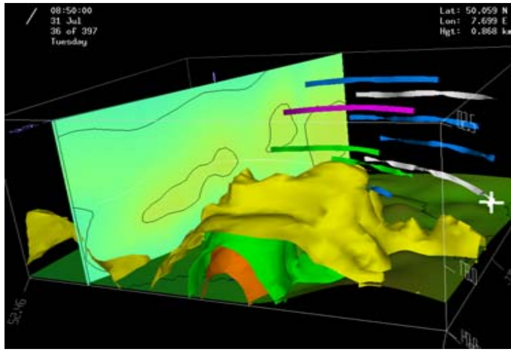
VIOLA-Applications (Multi-physics, Tele-collaboration)

- MetaTrace: Simulation of pollutant transport in groundwater with distributed SMP-Clusters (FZJ)
- TechSim: Distributed simulation of crystal growth and biosensors (Caesar)
- AMG-OPT: Parameter optimization and optimal algebraic solvers – Mechanical structure (SCAI)
- KoDaVis-Atmo: Collaborative visualization of huge atmospheric datasets (FZJ)









Fraunhofer Institut Algorithmen und Wissenschaftliches Rechnen

SCAI

Forschungszentrum Jülich
in der Helmholtz-Gemeinschaft

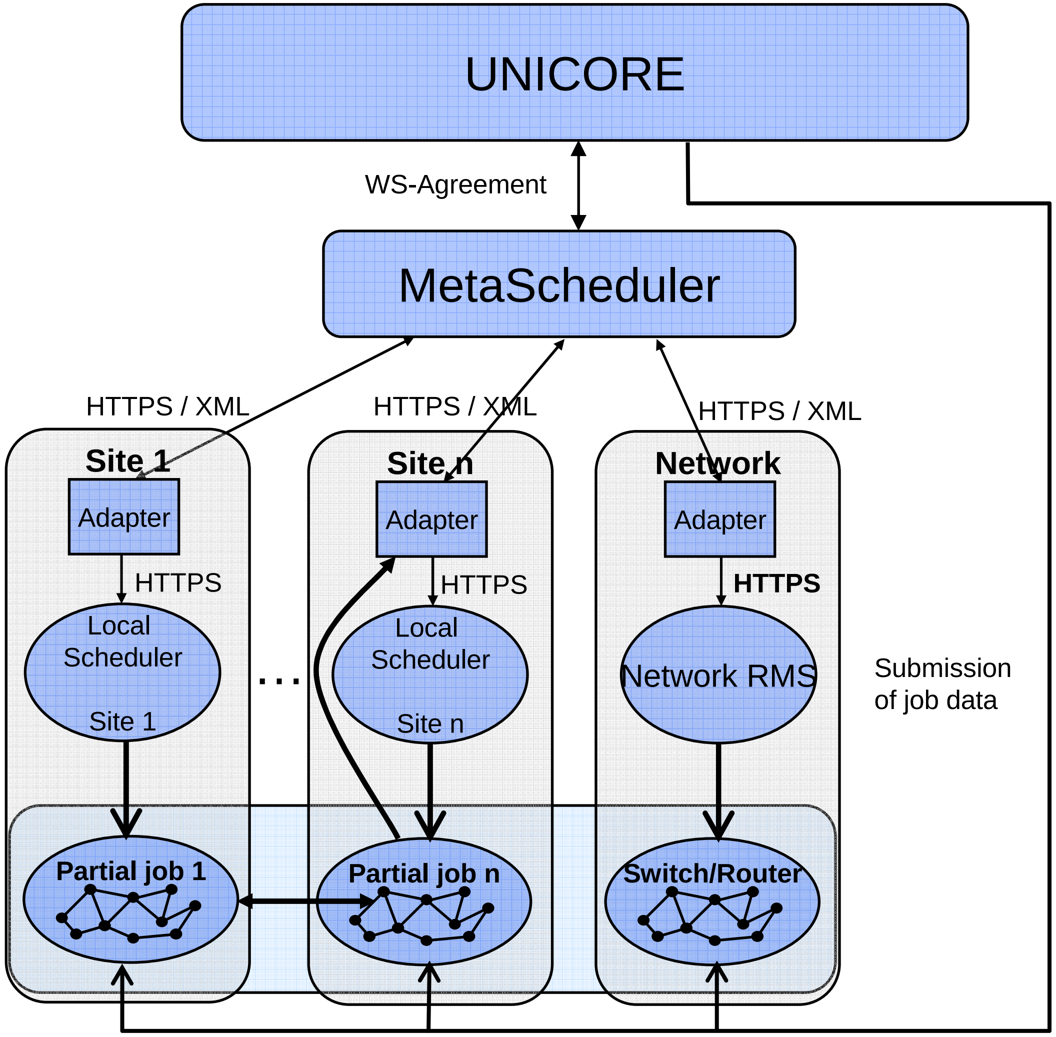
CoreGRID

VIOLA

The two Essential Properties of a local RMS

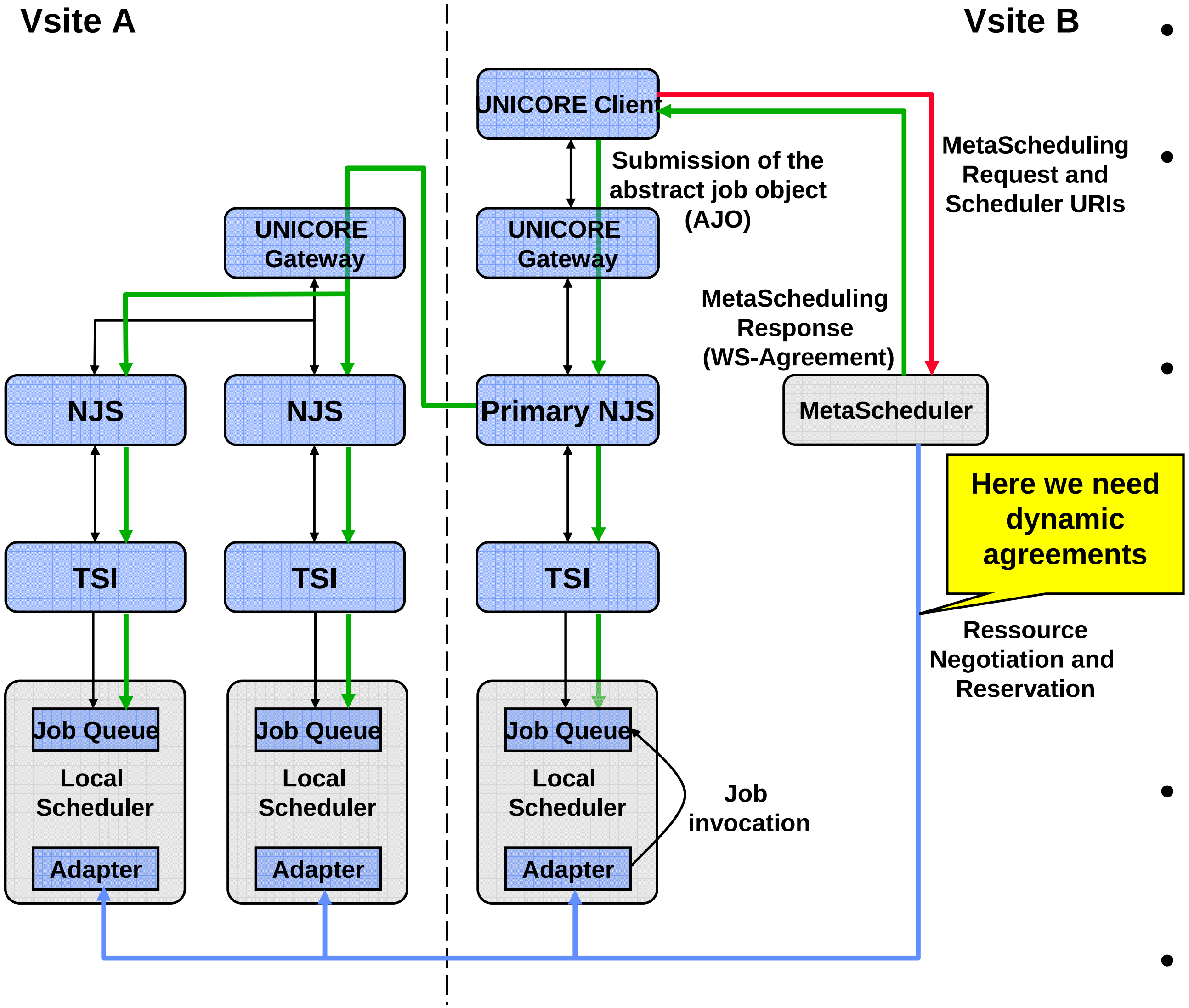
- Full backfill algorithm
- Estimation of worst case start/stop for each job (preview)
- Node range specification
- Start time specification (AT-jobs)
- Special resource requirement specification
- „very low priority“ jobs (Z-jobs)
- Communication friendly node allocation strategy
- Portable: available on different parallel machines
- Graphical user interface
- Status information available via WEB interface
- Priority scheme (project, resources, waited time)

MetaScheduler - Integration of local Schedulers



- Negotiation of timeslot & nodes with local schedulers for each job
- UNICORE initiates the reservation and submits the job-data
- UNICORE Client / MetaScheduler Service interface using WS-Agreement protocol
- Interface MetaScheduler / Adapters based on HTTPS/XML (SOAP)
- Interface between MetaScheduler Service and local RMS implemented with adapter pattern
- Authentification and Communication of Adapter and local Scheduler with ssh

MetaScheduler - Integration in UNICORE



- UNICORE Client sends request to MetaScheduler (WS-Agreement template)
- MetaScheduler negotiates earliest time to run this job, requests the reservation of the requested resources and returns the WS-Agreement with additional Status, ID
- UNICORE Client creates Abstract Job Object (AJO) and sends it to the Primary Network Job Supervisor (NJS)

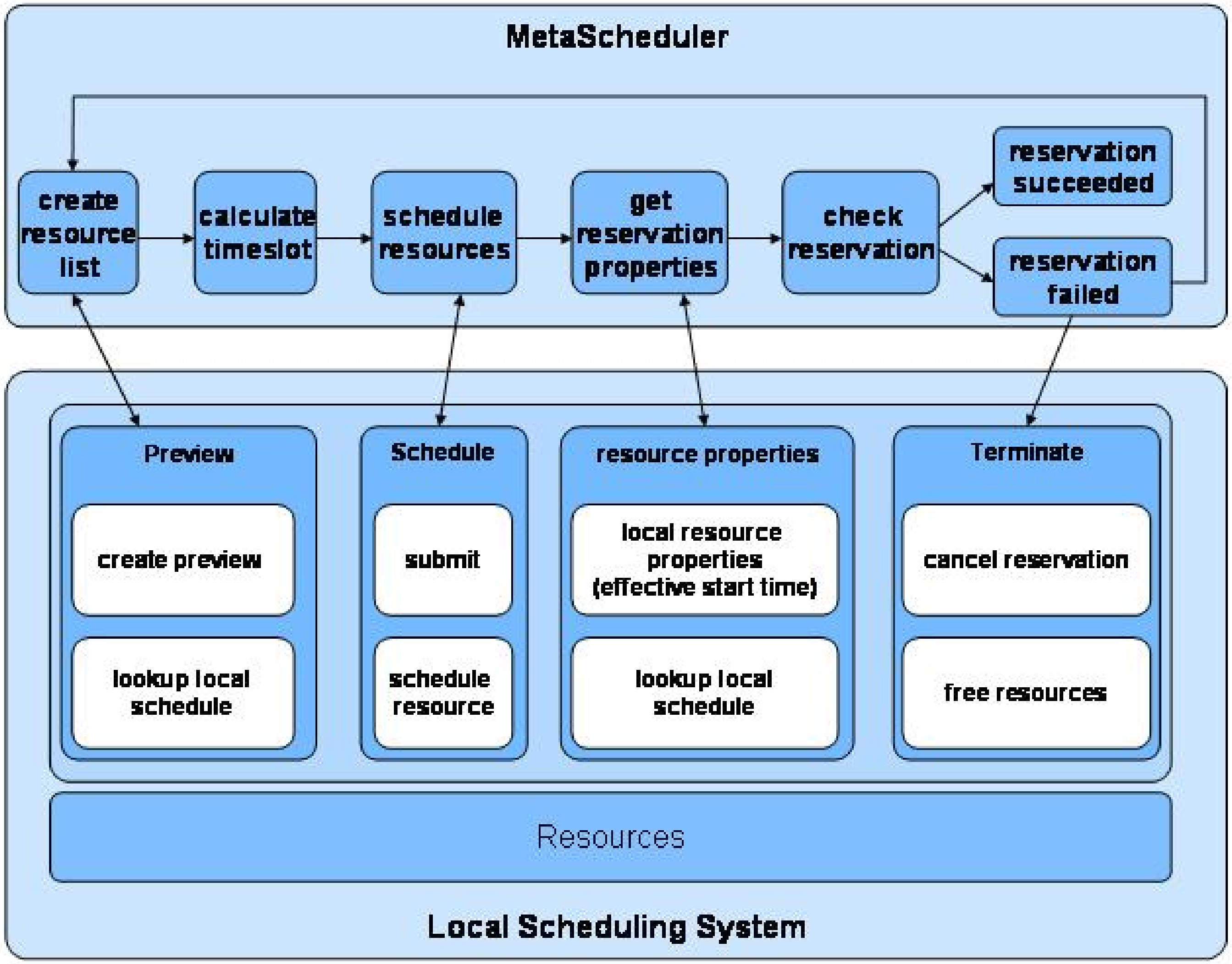
  NJS incarnates the AJO according to the information in the AJO and the UIDB, forwards it to the local Target System Interface (TSI) and sends the AJO to all other NJSs
- TSI creates the entry for the Meta-Job in the UNICORE Job Queue, and creates the UNICORE wrapper in the User-directory
- Scheduler triggers job at start time

Pseudo-code of the co-allocation algorithm

```
set  n                    =  number of requested resources
set  resources[1..n]      =  requested resources
set  properties[1..n]     =  requested property per resource   # number of nodes, bandwidth, time, ..
set  freeSlots[1..n]      =  null                                                     # start time of free slots
set  endOfPreviewWindow            =  false
set  nextStartupTime      =  currentTime+someMinutes          # the starting point when looking for free slots
while (endOfPreviewWindow = false) do {
        for 1..n do in parallel {
                freeSlots[i] = ResourceAvailableAt( resources[i], properties[i], nextStartupTime)
        }
        for 1..n do {
                set needNext = false
                if ( nextStartupTime != freeSlots[i]) then {
                        if ( freeSlots[i] != null) then {
                                if( nextStartupTime < freeSlots[i]) then {
                                        set nextStartupTime =  freeSlots[i]
                                        set needNext        =  true
                                }
                        } else {
                                set endOfPreviewWindow = true}
                }
        }
}

if ( ( needNext = false ) & ( endOfPreviewWindow = false) ) then return
        freeSlots[1] else return “no commont slot found”
```

Allocation Agreement Protocol



MetaScheduler

create resource list

calculate timeslot

schedule resources

get reservation properties

check reservation

reservation succeeded

reservation failed

Preview

create preview

lookup local schedule

Schedule

submit

schedule resource

resource properties

local resource properties (effective start time)

lookup local schedule

Terminate

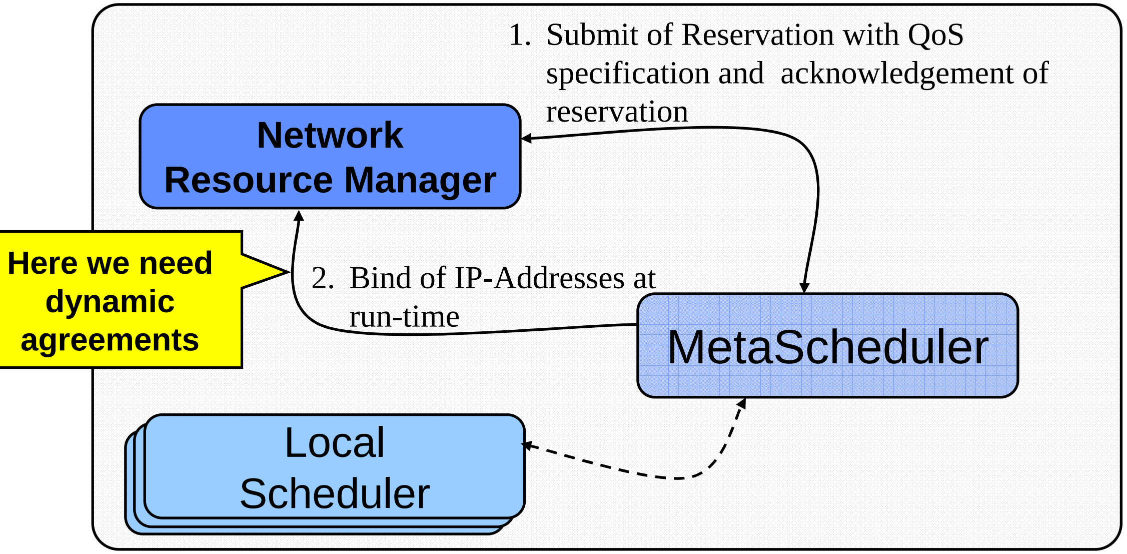
cancel reservation

free resources

Resources

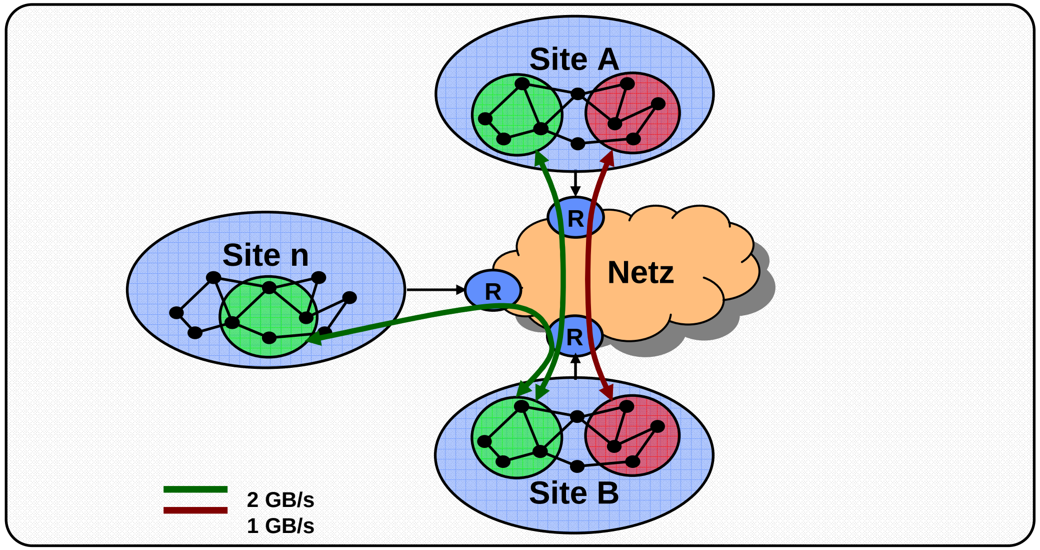
Local Scheduling System

Netzwork Resource Management System



1. Submit of Reservation with QoS specification and acknowledgement of reservation

2. Bind of IP-Addresses at run-time



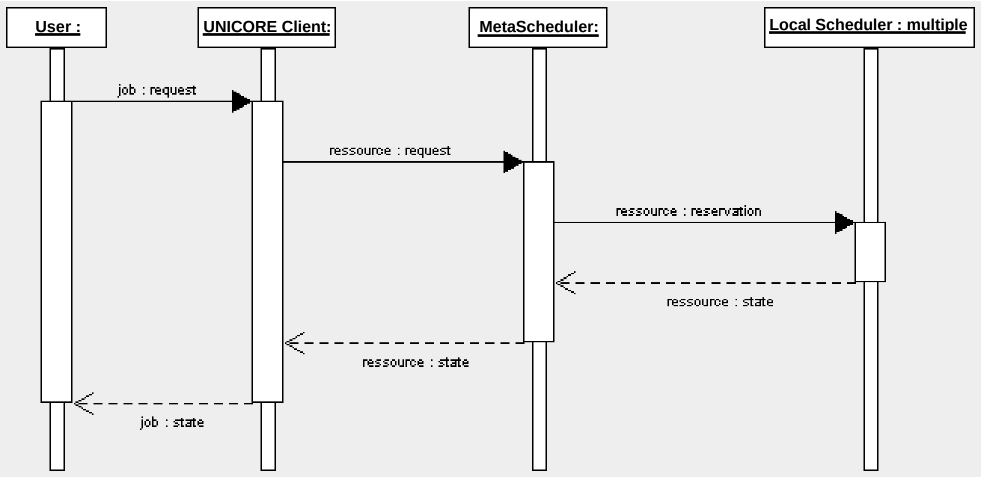
1.) Reservation of required Resources

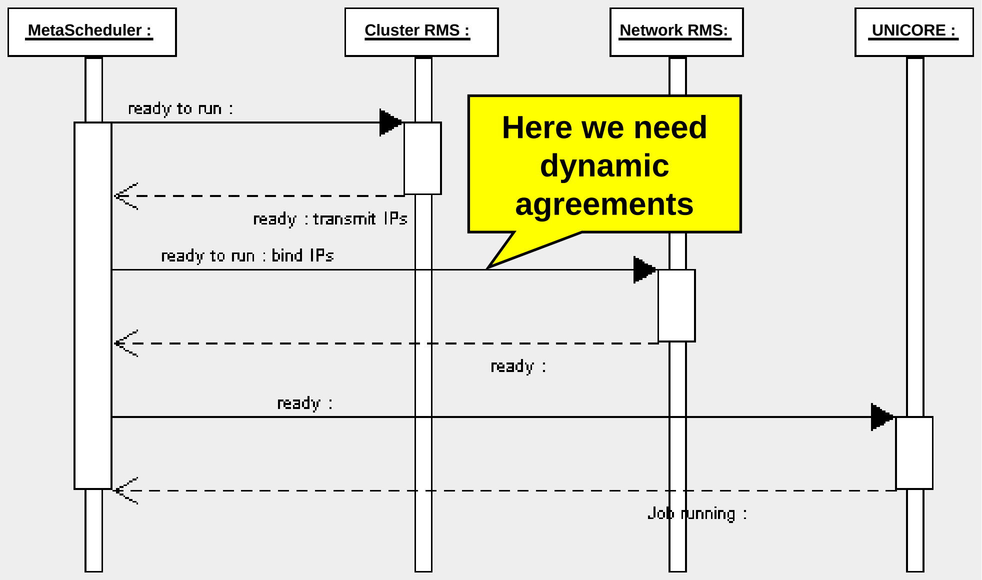
- Submit of a Reservation to the Network Resource Manager
- Acknowledgement of Reservation

2.) Bind of IP-Addresses at Run-time

- IP-Addresses are published at run-time of the job through the local Adapter
- Bind of the IP-Addresses by the Network Resource Manager
- Without explicit Bind the QoS Parameters for the Site-to-Site Interconnection are used

Scheduling of Network Resources





Here we need dynamic agreements

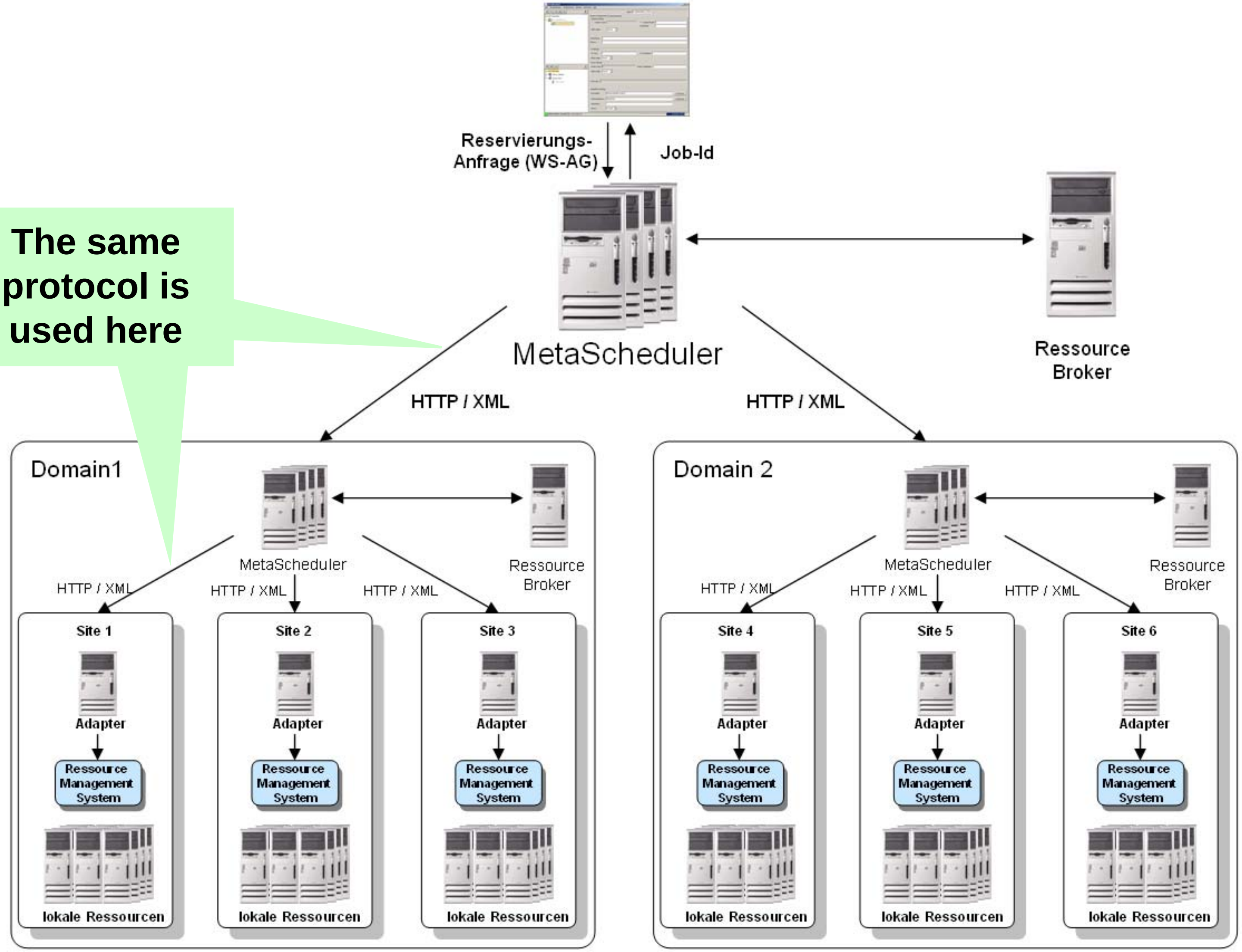
Reservation

- User describes Job Request with UNICORE Client
- Resolution of URIs of Resources und User names
- UNICORE client sends Resource Request to MetaScheduler (WS-Agreement template)
- MetaScheduler negotiates possible time for the Job and requests reservation of the Resources (Network, Nodes,…)
- MetaScheduler sends WS-Agreement with status of job and ID back to UNICORE Client

Start of Job

- At run-time the individual RMS know IP Addresses of local nodes that will be used for the job
- The MetaScheduler requests Status und IP Addresses from the local RMS and generates the global list of IP-Addresses
- Request of status of the Network RMS for the Job (ready to run)
- Dynamic Binding of IP-Addresses by Network RMS
- Generation of the global MetaMPI configuration and transfer to the individual sites
- Indicate status „Running“ to UNICORE (wrapper)

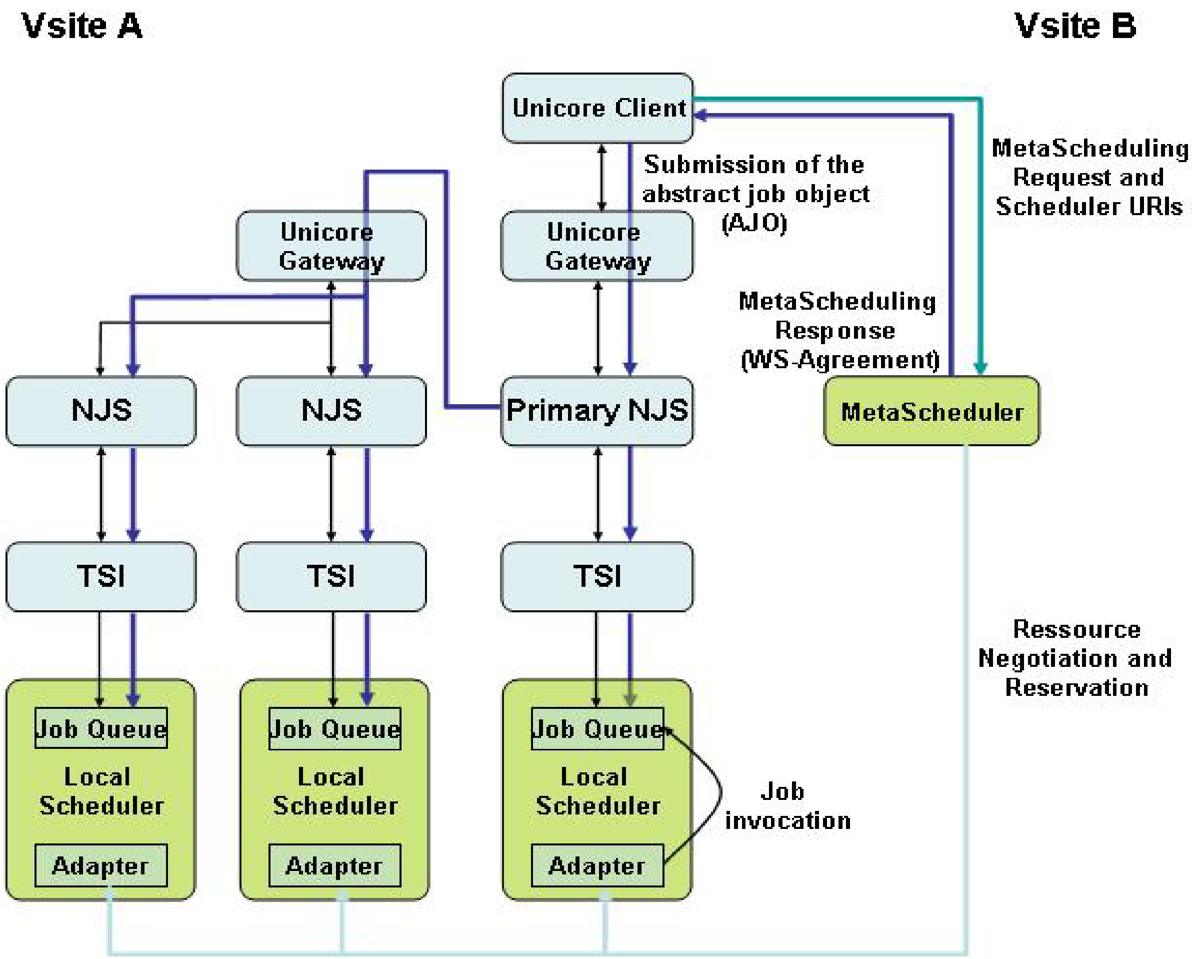
Inter-domain MetaScheduling



The same protocol is used here

Conclusion

- Co-Allocation of multiple Resources for distributed Applications in Grids needs flexible Agreements that may be modified instead of cancelled and re-established
- Agreements need to be changeable in order to includE information available only after the Agreement has been made
- We need a mechanism to federate individual related Agreements into one single Agreement
- We need a protocol to (re-) negotiate Agreements if necessary


Vsite A
Vsite B
Unicore Client
Submission of the abstract job object (AJO)
MetaScheduling Request and Scheduler URIs
Unicore Gateway
Unicore Gateway
MetaScheduling Response (WS-Agreement)
NJS
NJS
Primary NJS
MetaScheduler
TSI
TSI
TSI
Ressource Negotiation and Reservation
Job Queue
Local Scheduler
Adapter
Job Queue
Local Scheduler
Adapter
Job Queue
Local Scheduler
Adapter
Job invocation