Resource Management

Karl Czajkowski

ARGONNE NATIONAL LABORATORY
UNIVERSITY OF CHICAGO
epcc
KTH VETENSKAP OCH KONST
USC SCHOOL OF ENGINEERING

Prerequisites and Other Topics

- We assume some knowledge of:
  - Job submission systems
  - Resource management

  (We give only brief introduction.)
- See GRAM/Job session at GlobusWORLD
  - How to use GRAM tools
  - How to administer GRAM
  - How to program using GRAM APIs

  (We do not cover this today.)

Session Overview

Q: What is this session about?

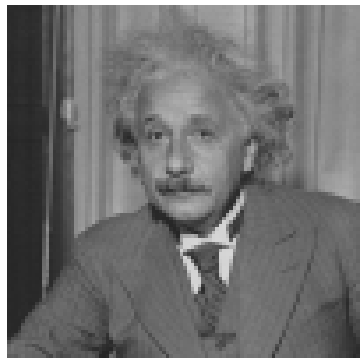
A: We will describe the world-view behind our Resource Management architecture, including our job management services and our standards efforts.

- Three-part discussion (~ 30 mins/each)
  - The nature of the Grid and our goals
  - Job management in GT 4.0
  - RM standards

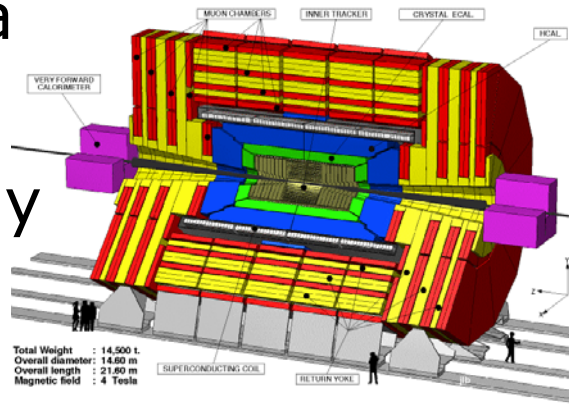
Resource Management: Part 1

Grid RM goals, future, and status…

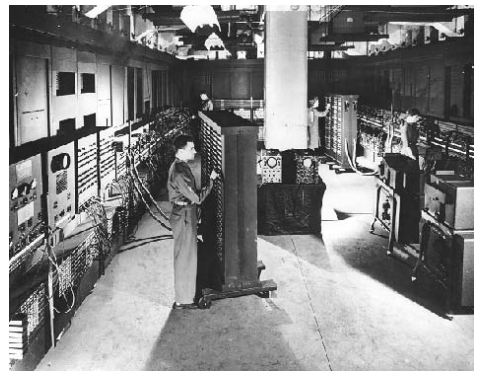
Why the Grid?
Origins: Revolution in Science



- Pre-Internet
  - Theorize &/or experiment, alone or in small teams; publish paper
- Post-Internet
  - Construct and mine large databases of observational or simulation data
  - Develop simulations & analyses
  - Access specialized devices remotely
  - Exchange information within distributed multidisciplinary teams



Why the Grid?
New Driver: Revolution in Business

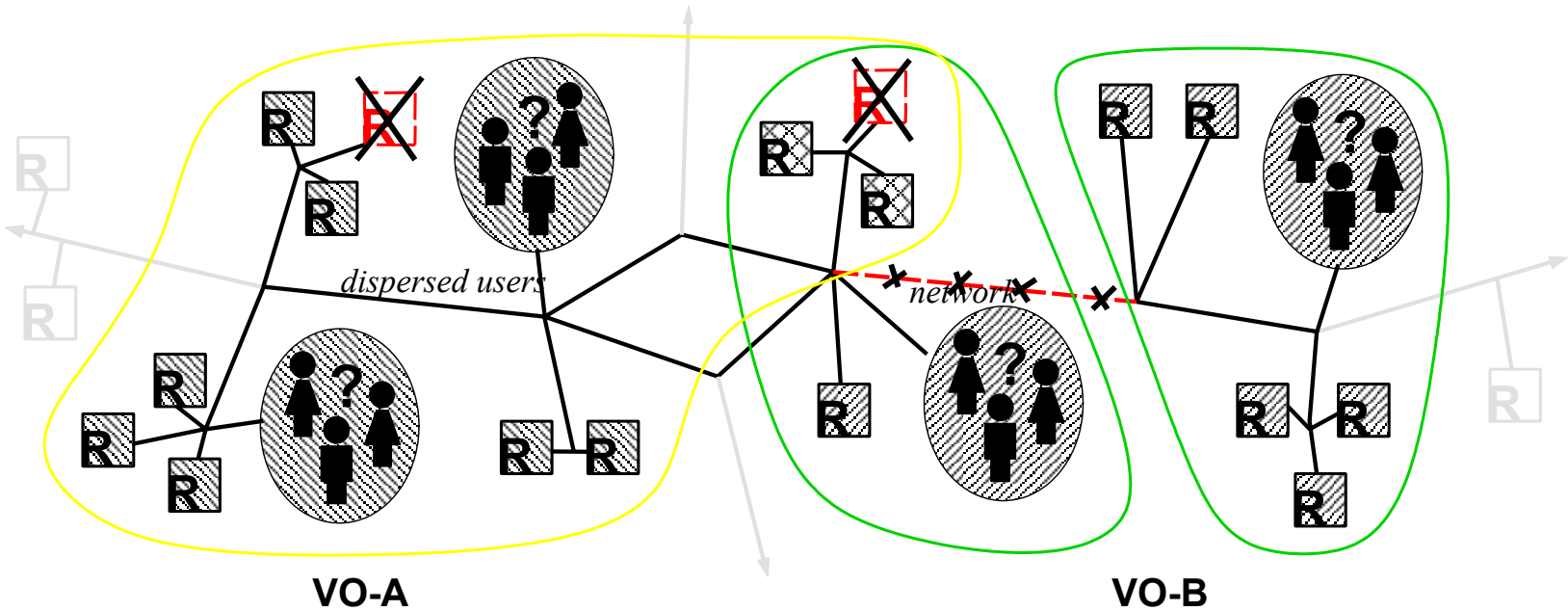


- Pre-Internet
  - Central data processing facility
- Post-Internet
  - Enterprise computing is highly distributed, heterogeneous, inter-enterprise (B2B)
  - Business processes increasingly computing- & data-rich
  - Outsourcing becomes feasible => service providers of various sorts
  - Growing complexity & need for more efficient management

Common eScience/eBusiness Requirements

- Dynamically link resources/services
  - From collaborators, customers, eUtilities, … (members of evolving "virtual organization")
- Into a "virtual computing system"
  - Dynamic, multi-faceted system spanning institutions and industries
  - Configured to meet instantaneous needs, for:
- Multi-faceted QoX for demanding workloads
  - Security, performance, reliability, etc.

Grid Resource Environment



- Distributed users and resources
- Variable resource status
- Variable grouping and connectivity
- Decentralized scheduling/policy

What Happens There?

- Data generation
  - Specialized instruments and sensors
  - Data processing and simulation
- Data storage
  - Bulk file storage and structured databases
  - Possible interface to instruments (buffering)
- Data movement
- Data processing
  - Compute data input to form output
  - Batch or interactive/coupled

Non-trivial Applications

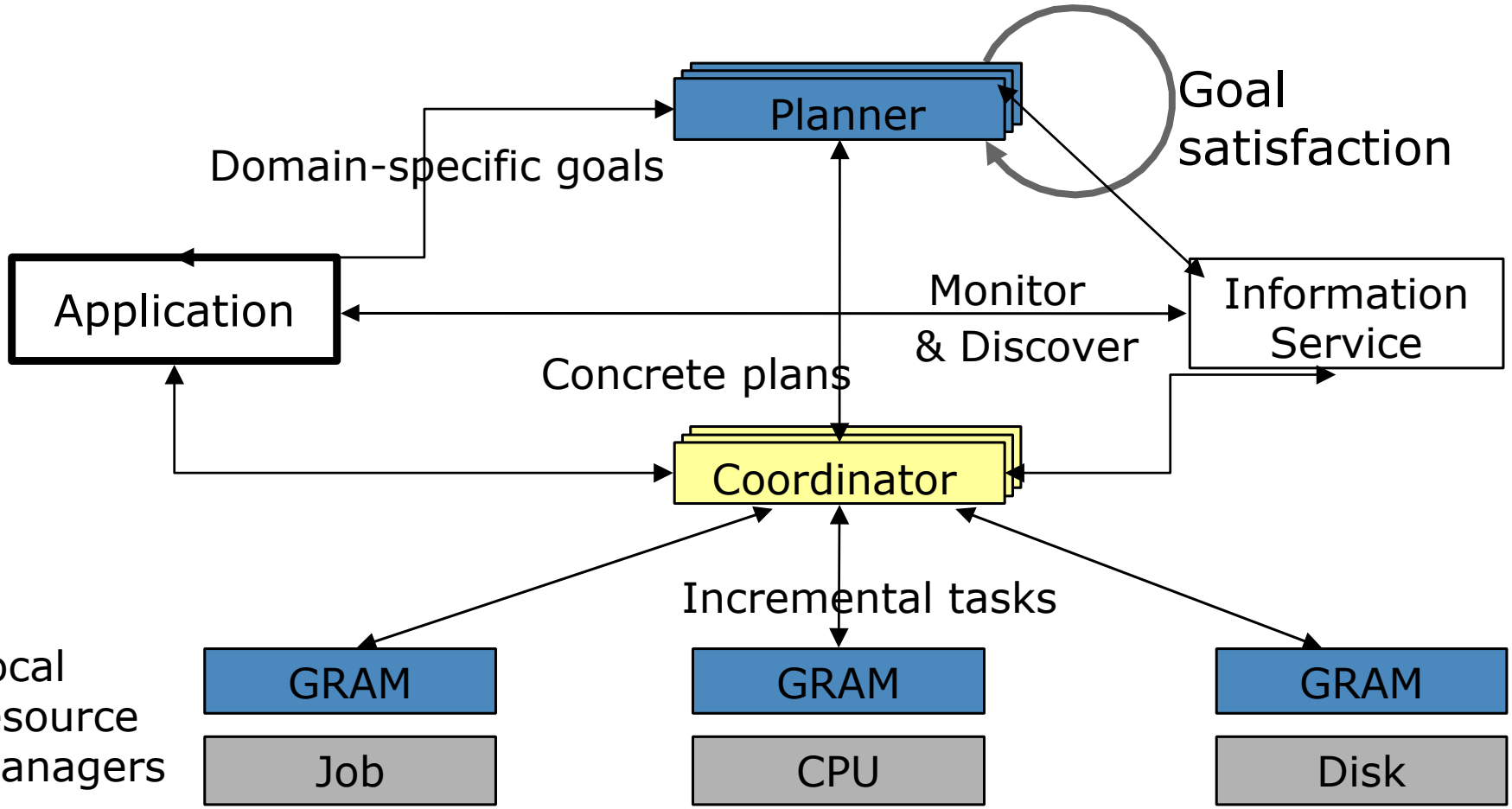
- Real-time or deadline-sensitive jobs
  - Wants localized, *very good* resource
- Large jobs
  - Large and/or coupled models
  - Wants to *coordinate* a few good resources
- High-throughput job sets
  - Many related jobs from one user/problem
  - Many unrelated jobs from many users
  - Wants scalable job control *everywhere*

Distributed Resource Management

1. Discovery
   - "What is out there? (of relevance) (to me)…"
   - Finds service providers
2. Inspection
   - "How do relevant providers compare?"
   - Compare policies, status, etc.
3. Agreement
   - "Will/did I get what I need?"
   - The core Resource Management problem

…Process can iterate due to adaptation

Long-term GRAM Architecture



Traditional Schedulers

- Closed-System Model
  - Presumption of global owner/authority
  - Sandboxed applications with no interactions
  - "Toss job over the fence and wait"
  - Emphasis on scheduler optimization
- Utilization as Primary Metric
  - Deep batch queues allow tighter packing
  - No incentives for matching user's schedule
- Sub-cultures Counter Site Policies
  - Users learn tricks for "gaming" their site

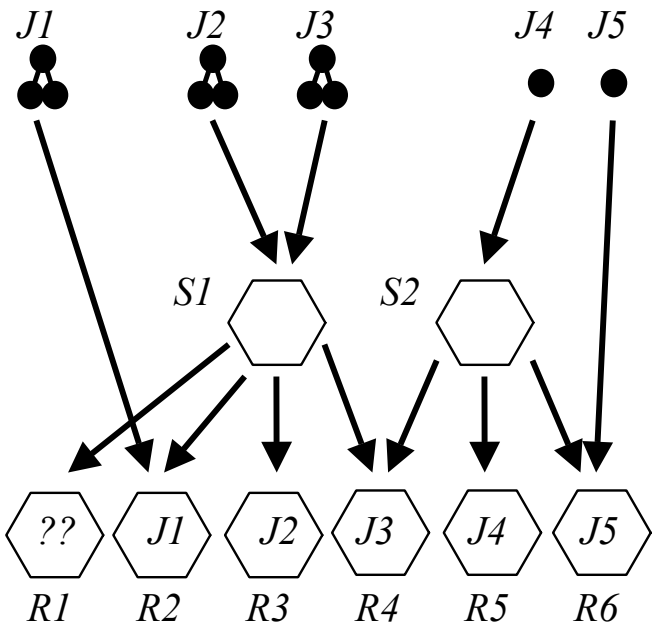
RM Mediates Conflict

- Resource Consumers/Applications Goals
  - Users: deadlines and availability goals
  - Applications: need coordinated resources
- Localized Resource Owner Goals
  - Policies distinguish users/communities
- Community Goals Emerge As:
  - Global optimization goals
  - Aggregate user, application and/or resource
- Reconcile demands via Agreement

An Open Negotiation Model

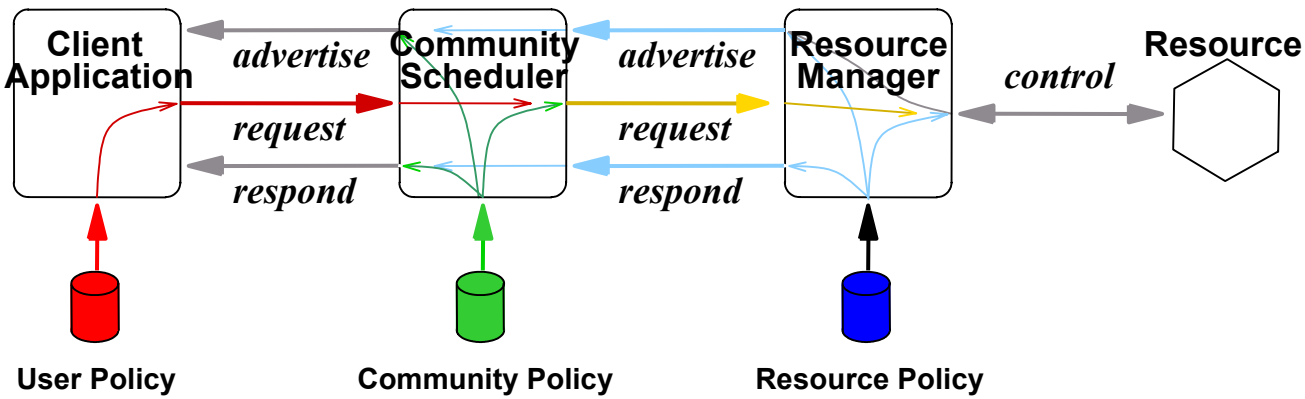
- Resources in a Global Context
  - Advertisement and negotiation
  - Normalized remote client interface
  - Resource maintains autonomy
- Users or Agents *Bridge* Resources
  - Drive task submission and provisioning
  - Coordinate acts across domains
- Community-based Mediation
  - Coordination for collective interest

Community Schedulers



- Individual users
  - Require service
  - Have application goals
- Community schedulers
  - Broker service
  - Aggregate scheduling
- Individual resources
  - Provide service to clients
  - Have policy autonomy

Intermediaries And Policy

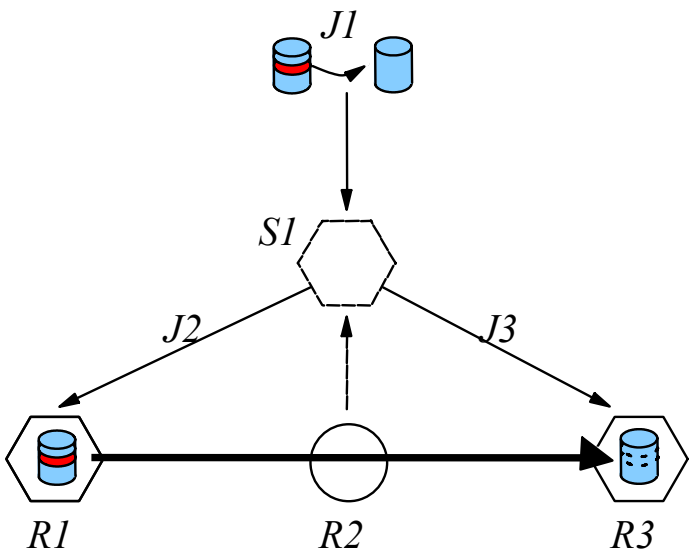


- Resource virtualization can:
  - Abstract details of underlying resource(s)
  - Abstract cardinality of aggregates
  - Map between different resource description domains
- Policies from different domains influence agreement negotiations with intermediaries

Heterogeneity of Service

- Many Kinds of "Task" or delivered service
  - Data: stored file, data read/write, packet xfer
  - Compute: execution, suspended job
- Many Kinds of Resource
  - Hardware: disks, CPU, memory, networks, display…
  - Capabilities: space, throughput…
- Coordination Problem is much the same
  - Difficulty varies
  - Applicability varies

Reliable File Transfer



- Single goal
  - Fast reliable transfer
- Specialized scheduler
  - Brokers basic services
  - Synthesizes new service
    - Fault-handling logic
- Virtualization
  - Aggregate manager
    - One request->many xfers
  - Abstracts gridFTP control

Technical Challenges

- Complex Security Requirements
  - Mixed identities, rights, audit, …
- Global Scalability
  - Similar ideals to Internet
  - Interoperable infrastructure
  - Policy-configurable for social needs
- Permanence or "Evolve in Place"
  - Cannot take World off-line for service
  - Over time: upgrade, extend, adapt

State of the Art

- Discovery is very hard and immature
  - Some viable information gathering systems
  - But information models have gaps
    - Lots of low-level "buttons and knobs" stuff, e.g. CIM
    - Some overly abstract stuff, e.g. GLUE, GRAM today
    - Complexity already a barrier to entry
  - RM policy: personalized scope/relevance
- Inspection is over-emphasized
  - Inherent race-conditions/scalability problems
- Basic allocation and "agreement" today
  - Implicit out-of-band intelligence still required

Future: Fuzzy

- Examples showed different tiers
- These manager tiers are illustrations
  - Not a claim of "proper" service topology
- VOs and application-specific managers
  - May have deeply "recursive" nesting
  - May form opaque virtualization barriers
    - Don't look behind the curtain
  - May form transparent brokers
    - Help resource providers and consumers to "meet"
  - Many front, back, and side doors for entry

Resource Management: Part 2

Job Management in GT 4.0

Job/Execution Duality

- GRAM supports job submission
  - A traditional "bare metal" job to run
  - With some data staging requirements
- GRAM supports execution management
  - User needs a virtual host/container
  - With some environment initialization
- Two sides of the same coin
  - All job submission IS resource virtualization
  - Some jobs more virtualizing than others!
    - Run a JVM? X Windows server? User-mode Linux?

Globus Toolkit GRAM Strategy

- Evolution via simultaneous deployment
  - Run multiple protocol engines per resource
  - Phase out old when satisfied w/ shift to new
- Refactoring abstract protocol
  - What do messages look like
- Refactoring GRAM protocol engine
  - How are messages processed
- Refactoring secure implementation
  - Multi-user job service w/ safety

Reasonable Applications Today

- High throughput job sets: two approaches
  1. Use GRAM for every task
     - The razor's edge of GRAM scalability
  2. Use GRAM for provisioning "slaves"
     - Course-grain jobs handle task/transaction flow
     - As in Condor glide-ins
- Large-scale jobs w/ MPICH-G3
  - Co-allocation but no co-reservation yet
- Special jobs
  - DIY discovery/control extensions

GRAM Protocol Evolution

- Traditional pre-WS GRAM
  - Custom RSL syntax, custom parser
  - Basic HTTP framing
- OGSI GRAM (dead end for standardization)
  - XML based, standard parser
- WS-GRAM
  - Simpler XML than OGSI GRAM had
  - WSRF consistent protocols
  - Direct approach, waiting for WS-Agreement

GRAM Implementation Evolution

- Pre-WS GRAM
  - Custom https engine
  - C state machine
  - perl “adapter” callouts
- WS-GRAM
  - Hosting environments' protocol engines
  - Java rewrite of state machine
  - Refined perl “adapter” callouts
    - Single adapter version can support both GRAM systems

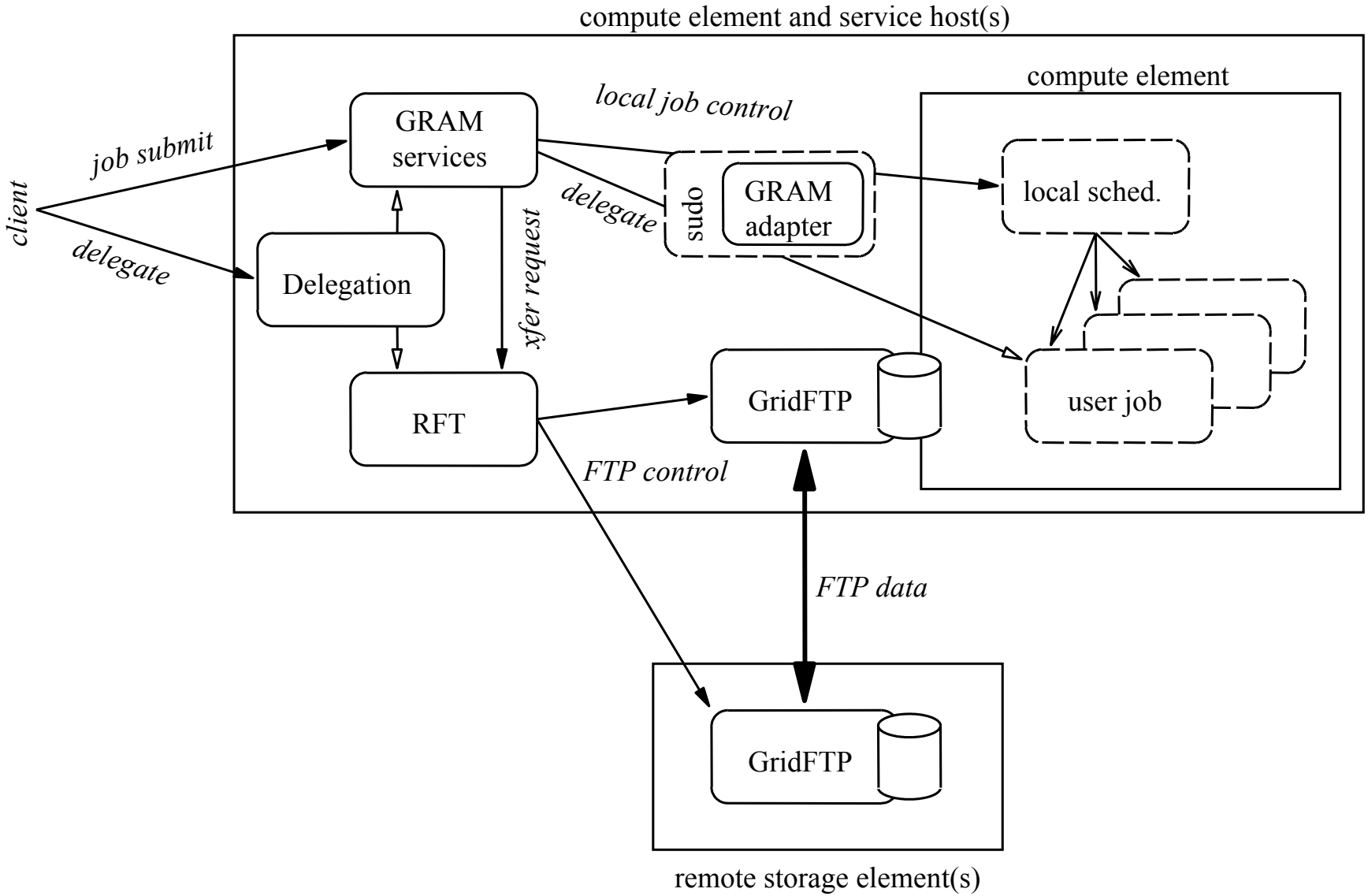
GRAM Security Evolution

- Pre-WS GRAM
  - gatekeeper https engine runs as root
  - job manager C and perl run as user
- OGSI GRAM
  - multiple hosting environments per host
  - too expensive w/ current Java WS tools
- WS-GRAM
  - non-root hosting process shared by all users
  - sudo callouts to GRAM user accounts
  - perl still runs as user

More WS-GRAM Improvements

- More efficient job monitoring
  - Signal-based fork job monitoring
  - Log-based queueing system monitoring
  - No more expensive/slow polling
  - No more frail queue snapshot comparators
- Refactored optional components
  - Staging of files via RFT+gridftp
  - Explicit delegated credential management

WS-GRAM Approach



Staging via RFT+gridFTP

- Simplified job state machine impl.
- More transfer reliability and performance
  - Better behaviour than naïve "GASS" library
  - Benefit from new gridFTP servers over time
- Streaming access: efficient client poll
  - Applies to any output, not just stdout/stderr
  - More robust restart/recovery for client
  - Introduces new namespace wrinkles

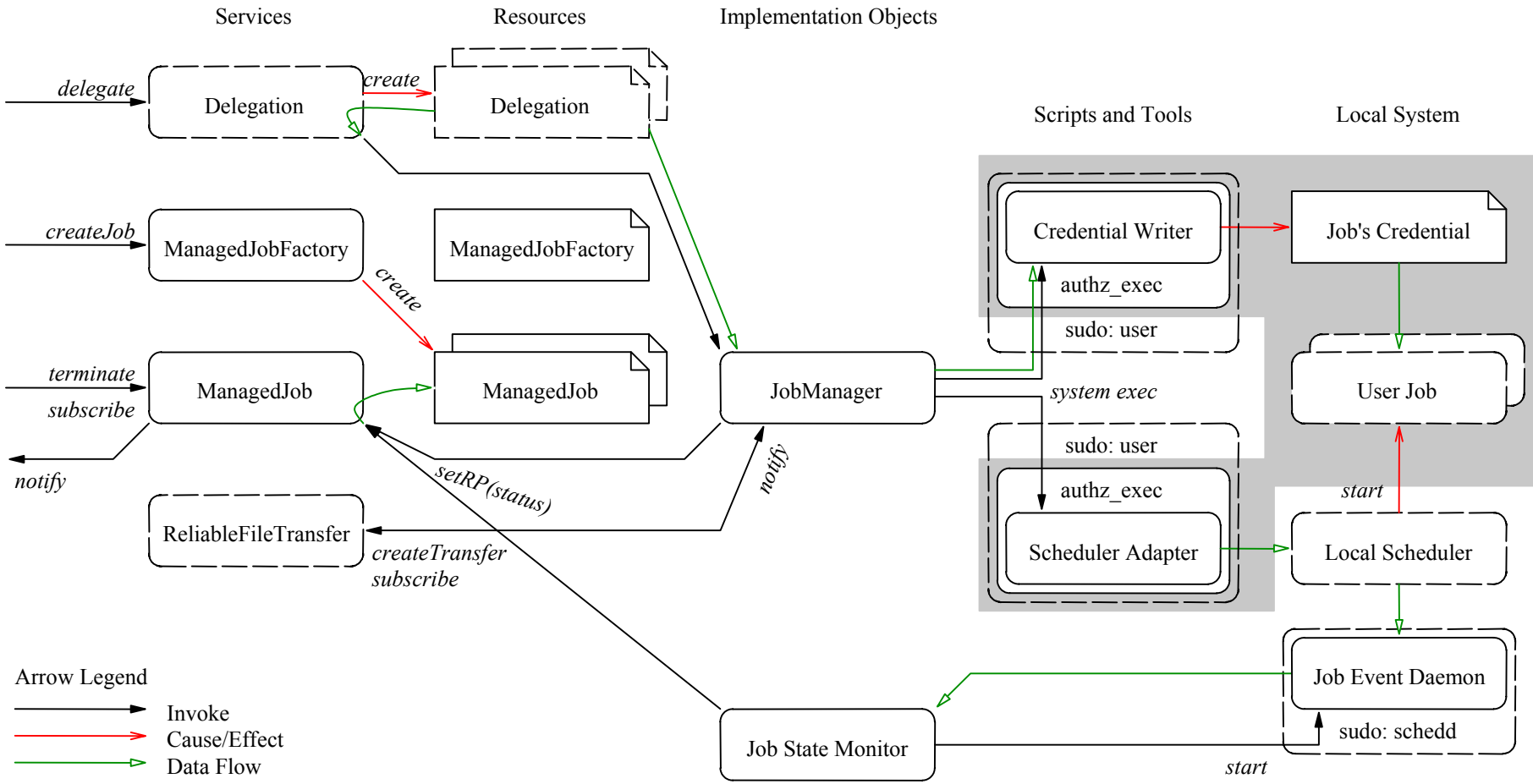
Explicit Credential Management

- Client-controlled delegation
  - Completely optional
  - Customize for job/user requirements
- Supports reuse
  - Delegate once, use w/ multiple jobs
- Efficient and understandable
  - User can refresh once per remote host
    - Updates all jobs sharing that credential
  - User can choose to separate
    - For jobs or job sets w/ differing trust or lifecycles

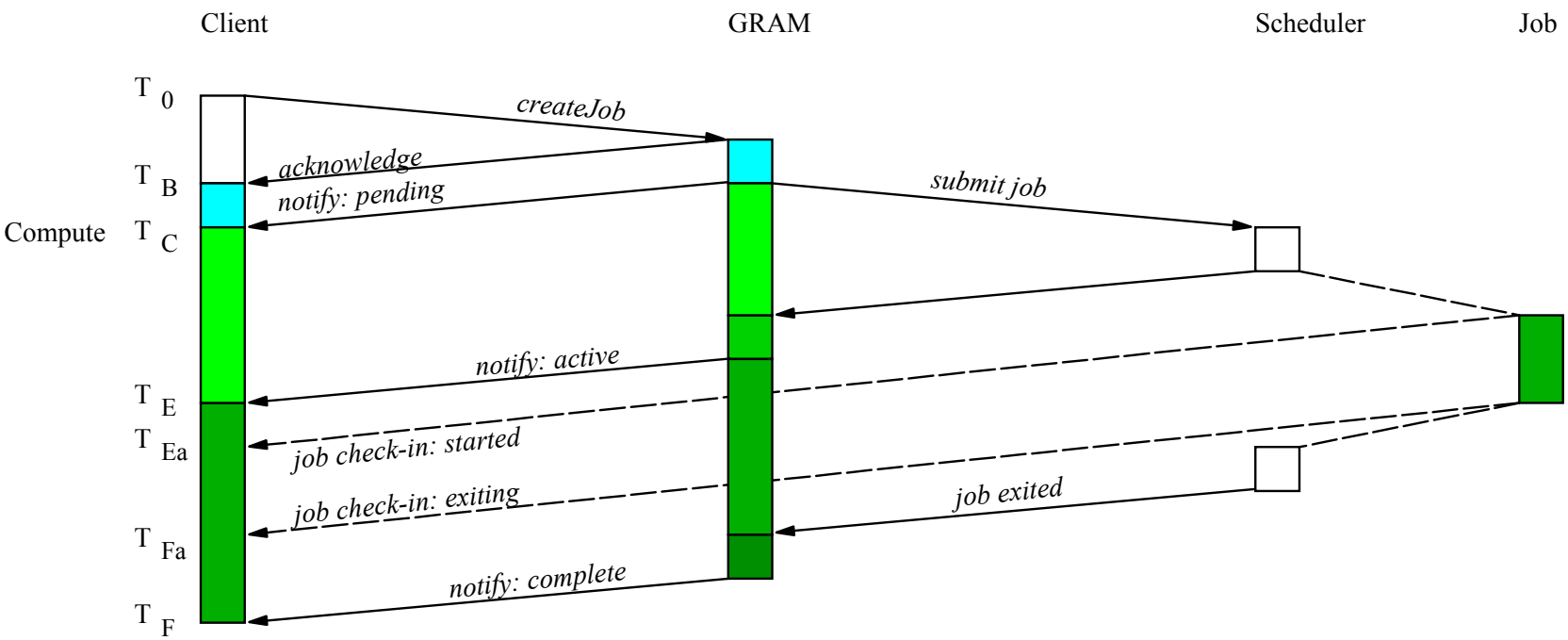
Performance Improvements

- Scalability better than GT3.x
  - One hosting environment instead of N
  - Less impact on queueing system (no polling)
  - Robust staging of large filesets w/ RFT
- More improvements in Java XML processing
  - Lower CPU overhead
  - Better memory footprint
- C-based client tool and libs
  - Low CPU overhead compared to Java
  - Low latency command-line tool startup

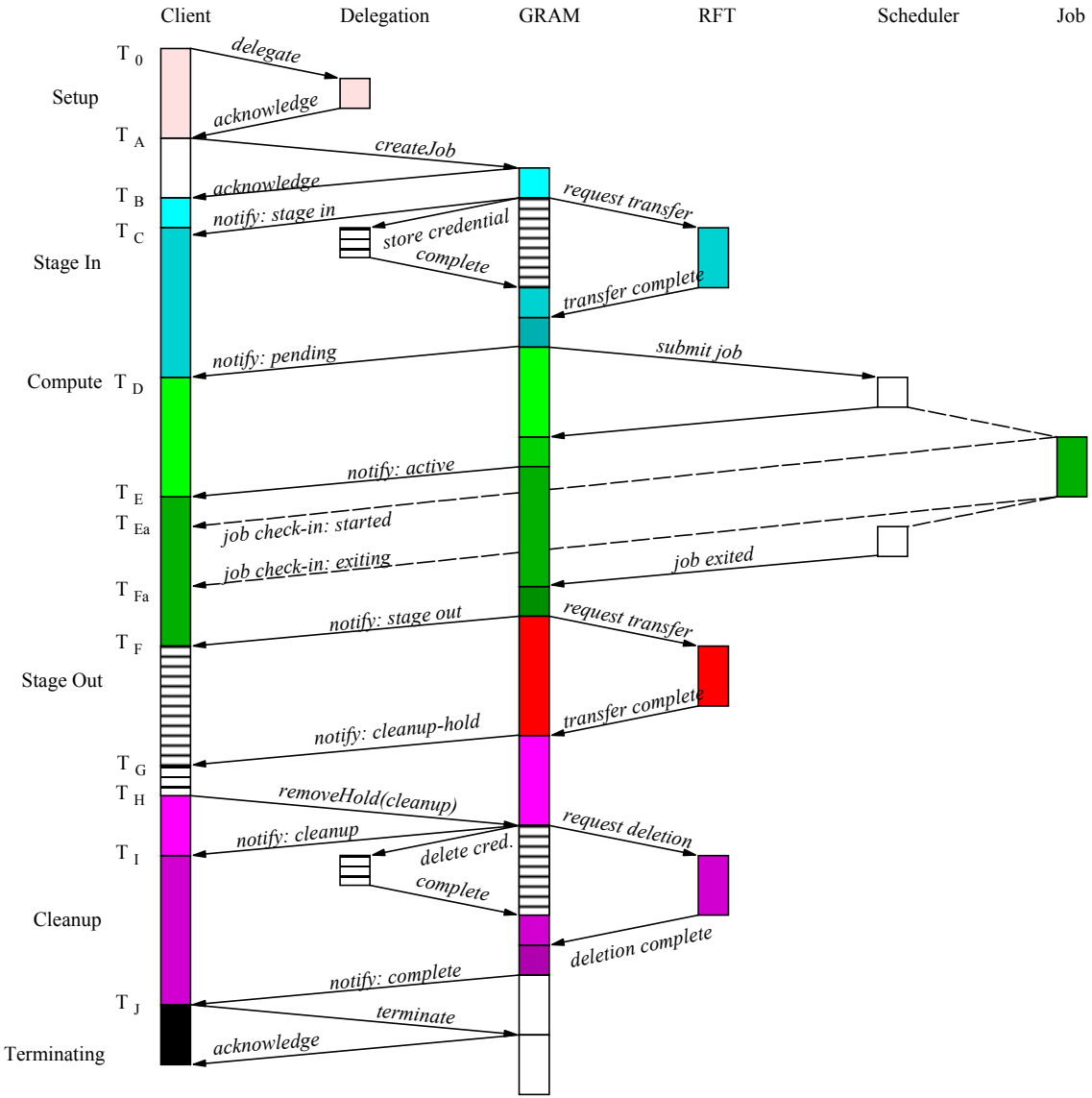
WS-GRAM Software Map



WS-GRAM Base Protocol



WS-GRAM Full Protocol



Positioning for Future

- Better modularity of job manager functions
  - Track improvements in RFT, gridFTP
  - Experimentation w/ other job protocols
    - Share sudo/perl callouts for job control
    - Provide different "views" of job execution system
  - Reuse functions for next job standard
- Higher-level WSRF programming model
  - Return of co-allocation for MPICH
  - Unconventional job-like services next?
  - Advance reservation or co-scheduling next?

Co-Allocation for MPICH

- Comparable to pre-WS DUROC component
  - But server-side instead of client-side
- Multi-job manager
  - Same interface pattern as for regular job
  - Manages multi-job aggregation
    - Recursively split and run regular job submits
- Rendezvous support
  - Intra-subjob task coordination (a la gram_myjob)
  - Subjob coordination (a la DUROC barrier/comm.)

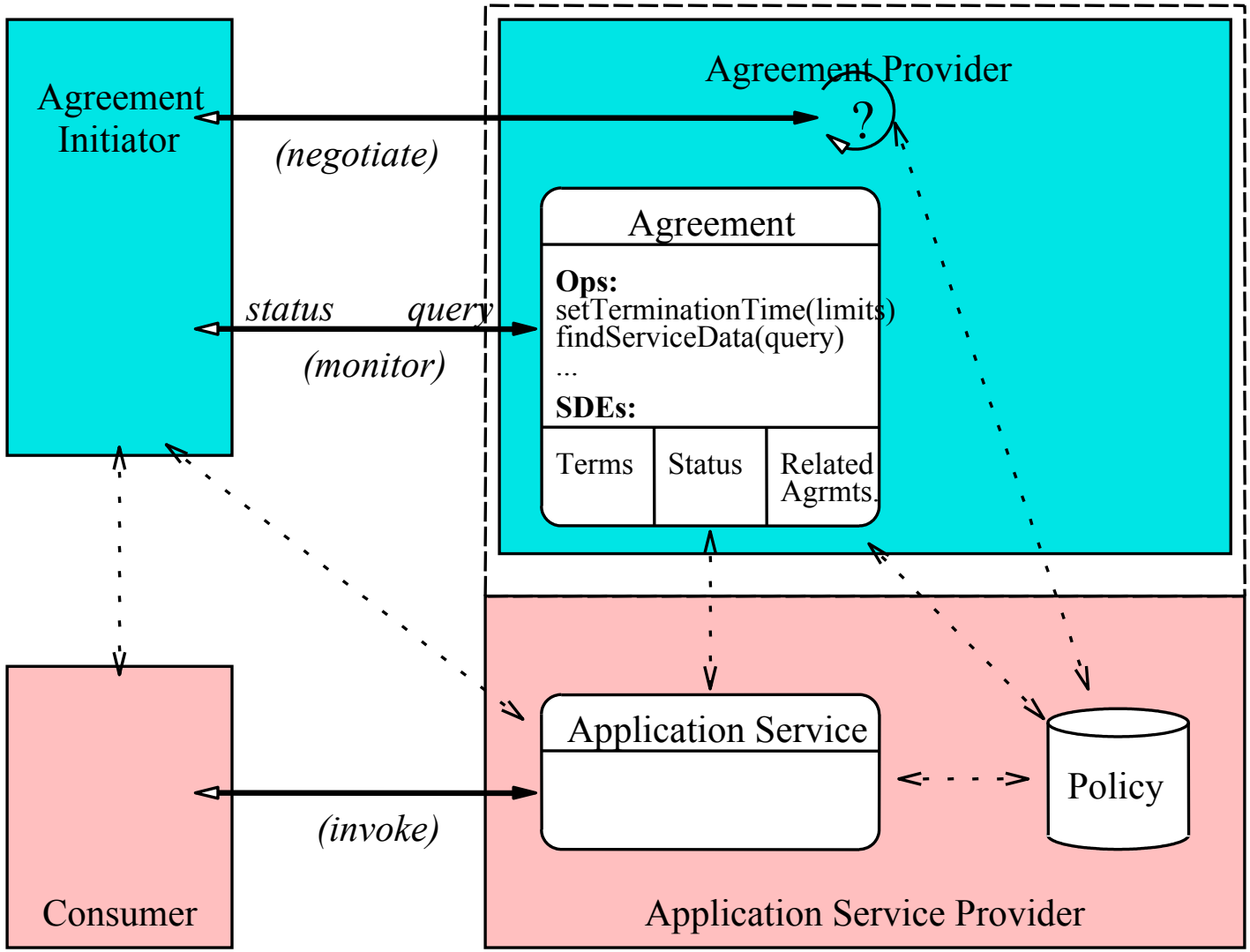
Resource Management: Part 3

WS-Agreement and the future…

WS-Agreement

- New standardization effort
  - In GGF's GRAAP-WG
  - In public comment period
- Generalizes GRAM ideas
  - Service-oriented architecture
  - Resource becomes *Service Provider*
  - Tasks become *Negotiated Services*
  - State presented as *Agreement* services
- Supports extensible domain terms

WS-Agreement Entities



Simple Negotiation

- AgreementFactory::createAgreement()
  - Coarse-grained
  - Conventional fault/response model
  - Batch negotiation of complex terms
  - Idiom: enables one-shot job submission
- Agreements can be chained
  - Establish stateful context of Agreements
  - New Agreement depends on/claims context
- Need companion specs for advanced scenarios

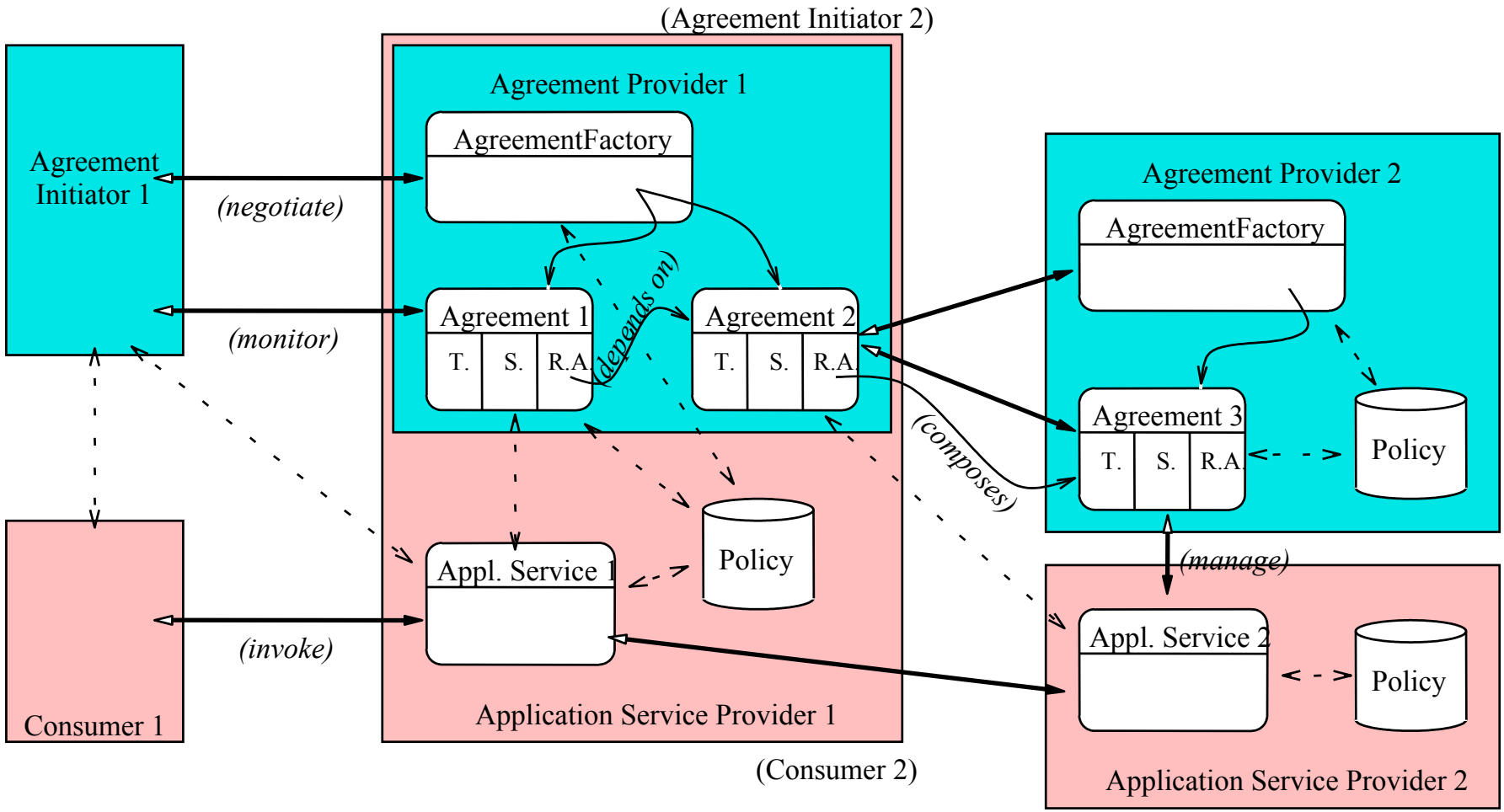
Agreement-based Jobs

- Agreement represents “queue entry”
  - Commitment with job parameters etc.
- Agreement Provider
  - i.e. Job scheduler/Queuing system
  - Management interface to service provider
- Service Provider
  - i.e. scheduled resource (compute nodes)
- Provided Service is the Job computation

Advance Reservation for Jobs

- Schedule-based commitment of service
  - Requires schedule based Agreement terms
- Optional Pre-Agreement
  - Agreement to facilitate future Job Agreement
  - Characterizes virtual resource needed for Job
  - May not need full job terms
- Job Agreement almost as usual
  - May exploit Pre-Agreement, or
    - Reference existing promise of resource schedule
  - May get schedule commitment in one shot

Virtualization and Brokers



Example Use in Enterprise: SAP Demonstration @ TechEd

- SAP demo of 3 Globus-enabled applications:
  - CRM: Internet Pricing & Configurator (IPC)
  - CRM: Workforce Management (WFM)
  - SCM: Advanced Planner & Optimizer (APO)
- Applications modified to:
  - Adjust to varying demand and resources
  - Use Globus to discover and provision resources
- Resulting in:
  - Lower TCO: Fewer servers & ease of use
  - Performance improvement

SAP Grid IPC Overview

Web Browsers / Batch Processes

(typically several thousand requests)

Request:
Price Query

1

IPC Dispatcher

2

IPC Server

Delegation of Request

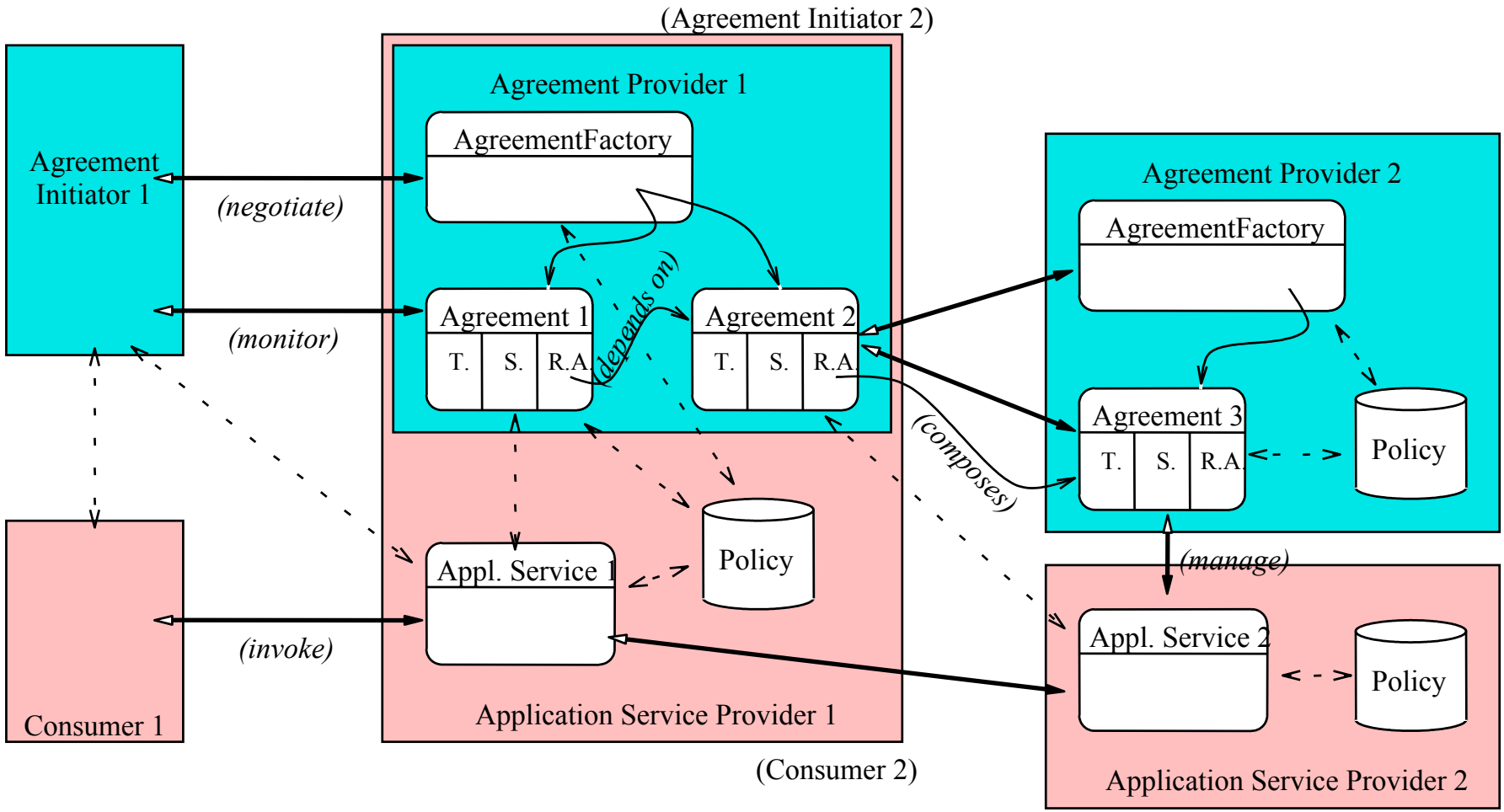
2

IPC Server

Response: Pricelist
Depending on:
- Time
- Discount
- Number of Items
- …

3

Virtualized Providers



Future Models

- Service behavioral descriptions
  - Unified service term model
  - Capture user/application requirements
  - Capture provider capabilities
- Core meta-language
  - Facilitates planner/decision designs
  - Extends with domain concepts
  - Extensible negotiability mark-up
    - Assist discovery/brokering
    - Capture range of negotiability for variable terms
    - Capture importance of terms (required/optional)

Related Work

- Academic Contemporaries
  - Condor Matchmaking
  - Economy-based Scheduling
  - Work-flow Planning
- Commercial Scheduler Examples
  - CSF (Community Scheduler Framework)
    - Demonstrating WS-Agreement intermediary
  - Many examples for traditional sites
    - Expose through GRAM or native WS-Agreement support?
    - Platform Computing
      - LSF scales to lots of jobs
      - MultiCluster for site-to-site resource sharing
  - Work-flow/transaction managers

WS-Agreement is a Protocol

- WS-Agreement is a message model…
  - Not a component
- …applicable to previous examples
  - Interface standard between components
  - Improve interoperability of other systems
  - To enable composition/federation

(Previous examples:

  - GRAM, Condor
  - Workflow, economic scheduling
  - PBS, LSF, CSF)

Specifying Terms: Who and What?

In a service provisioning **domain**
(e.g. "computational jobs")

- A **standard** specifies **domain** terms
- A **provider** specifies its support for
  - *some or all* of the domain standard terms
  - a given term, *specifically*
    - Within *behavioral* constraints
    - Within *negotiability* constraints
    - With extra fields/sub-terms
    - Arbitrary term *properties*: e.g. *optional* or *required*
- *A **client** discovers compatible providers*

Possible GRAM Avenues

- Choose a job term language (e.g. JSDL?)
  - Don't rush for yet another job dialect
  - Define a profile for terms+mark-up?
- Provide an AgreementFactory impl.
- Provide an Agreement impl.
- Tie into existing ManagedJob resource impl.
- Support both WS-GRAM, Job Agreements
  - Get experience without “betting the farm”
  - Ease migration

Conclusion

- GRAM development is evolutionary
  - Following changes in service environment
  - Improving interaction with schedulers
  - Improving file staging behavior
- Job management is refactored
  - Better separation from protocol engine
- The future is getting closer
  - WS-Agreement
  - RM for more than just job submission
- We want to support smooth migration