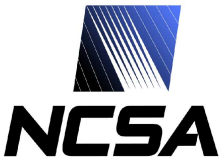
Basic Execution Management input from Globus Alliance

Karl Czajkowski

Univa Corporation



Must Support: 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 Execution Management

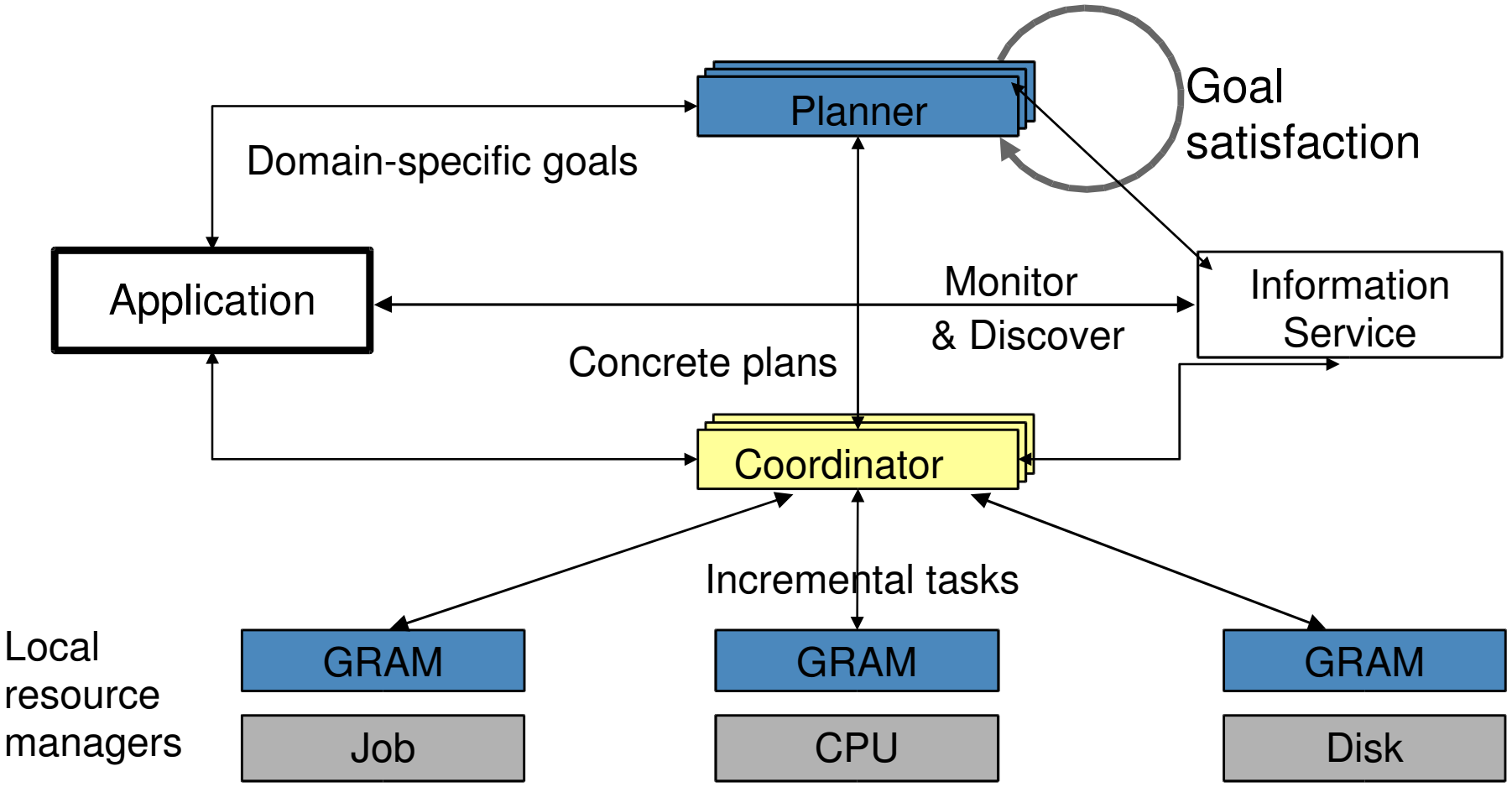
1. Discovery
   - "What is out there? (of relevance) (to me)…"
   - Finds BES providers
2. Inspection
   - "How do relevant providers compare?"
   - Compare policies, status, etc.
3. Agreement
   - "Will/did I get what I need executed?"
   - The core Execution Management problem

…Process can iterate due to adaptation

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?

Long-term GRAM Architecture



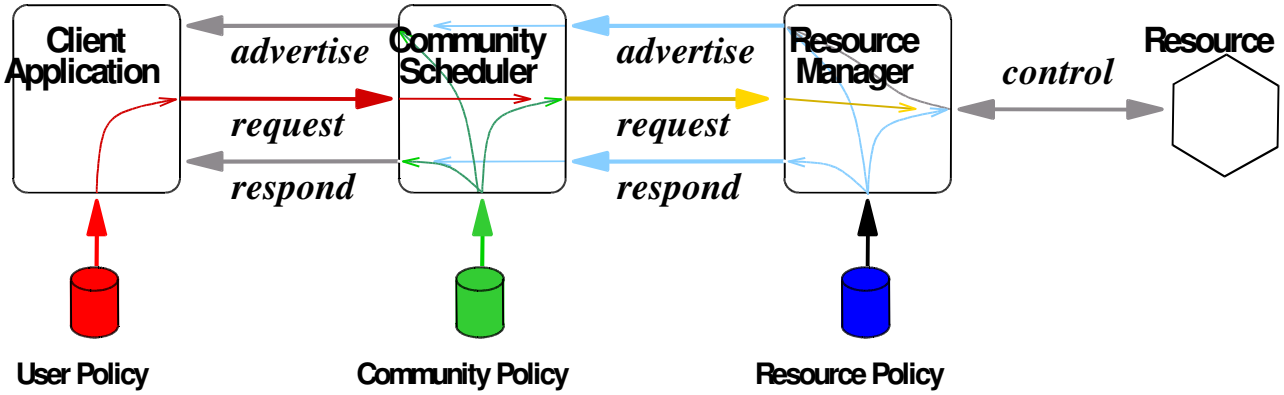
Exec. Mgmt. Mediates Conflict

- 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

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

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

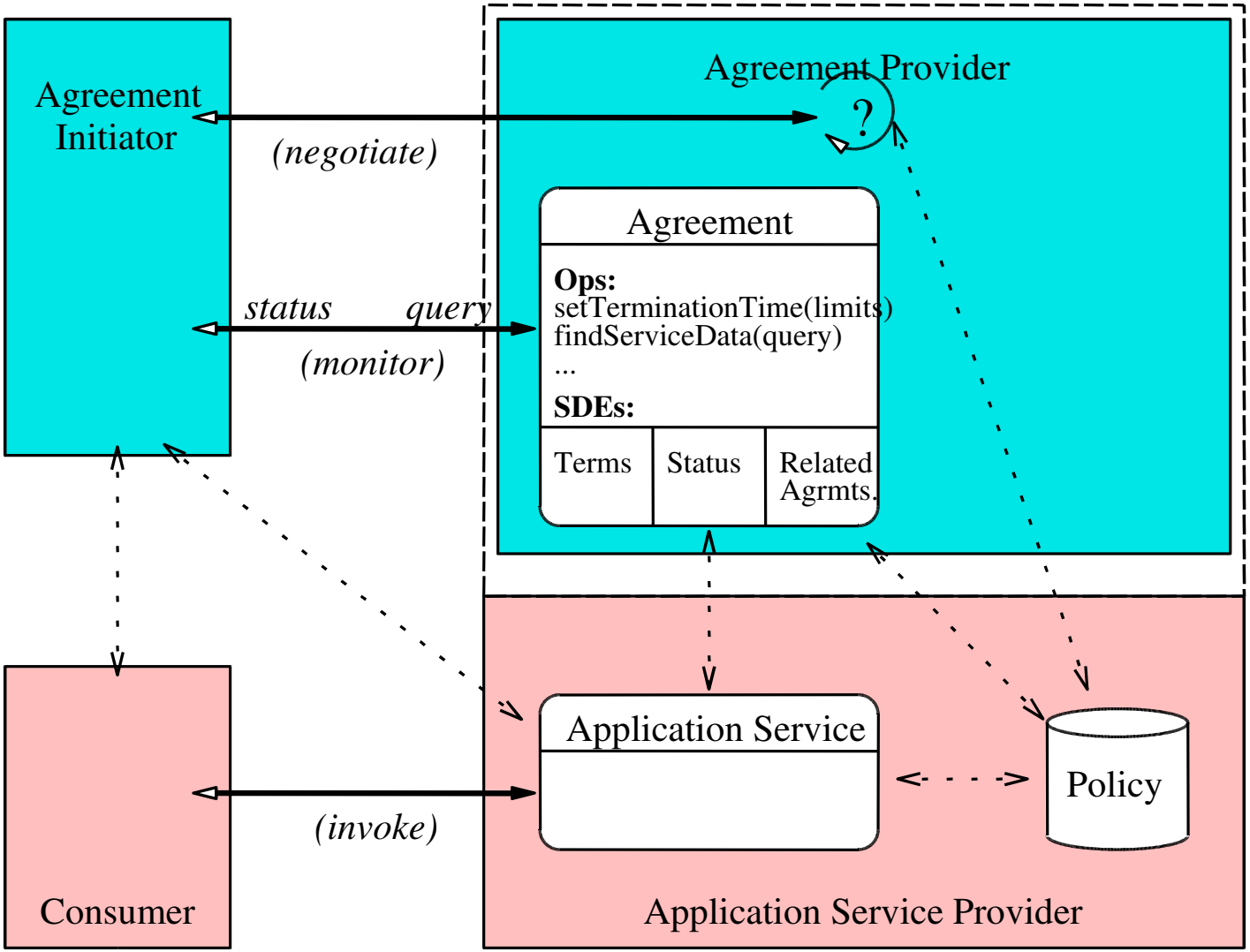
State of the Art

- 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

WS-Agreement

- Ongoing standardization effort
  - In GGF's GRAAP-WG
  - Several issues raised in public comment period
    - Need 3-6 months to address and reenter public comment?
- Generalizes GRAM ideas
  - Service-oriented architecture
  - Resource becomes *Service Provider*
  - Tasks become *Negotiated Services*
  - State presented as *Agreement* services
- Supports composition w/ 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.
  - Management interface to dynamic Job(s)
- 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

WS-Agreement is a Protocol

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

(Possible WS-Agreement conversion examples:

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

Specifying Terms: Who and What?

In a service provisioning **domain**

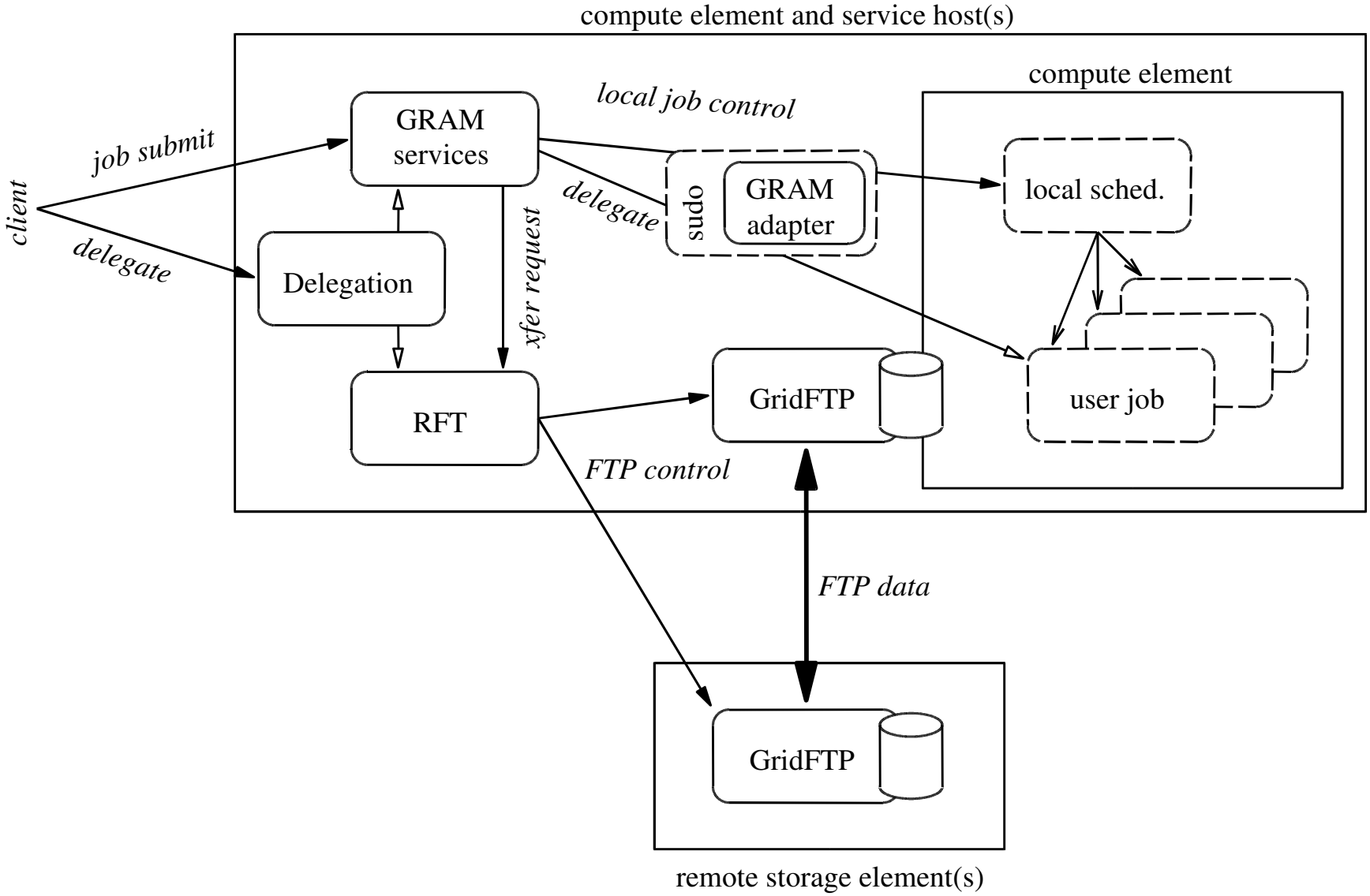
(e.g. “computational jobs”)

- A **standard** specifies **domain** terms/concepts
- 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*

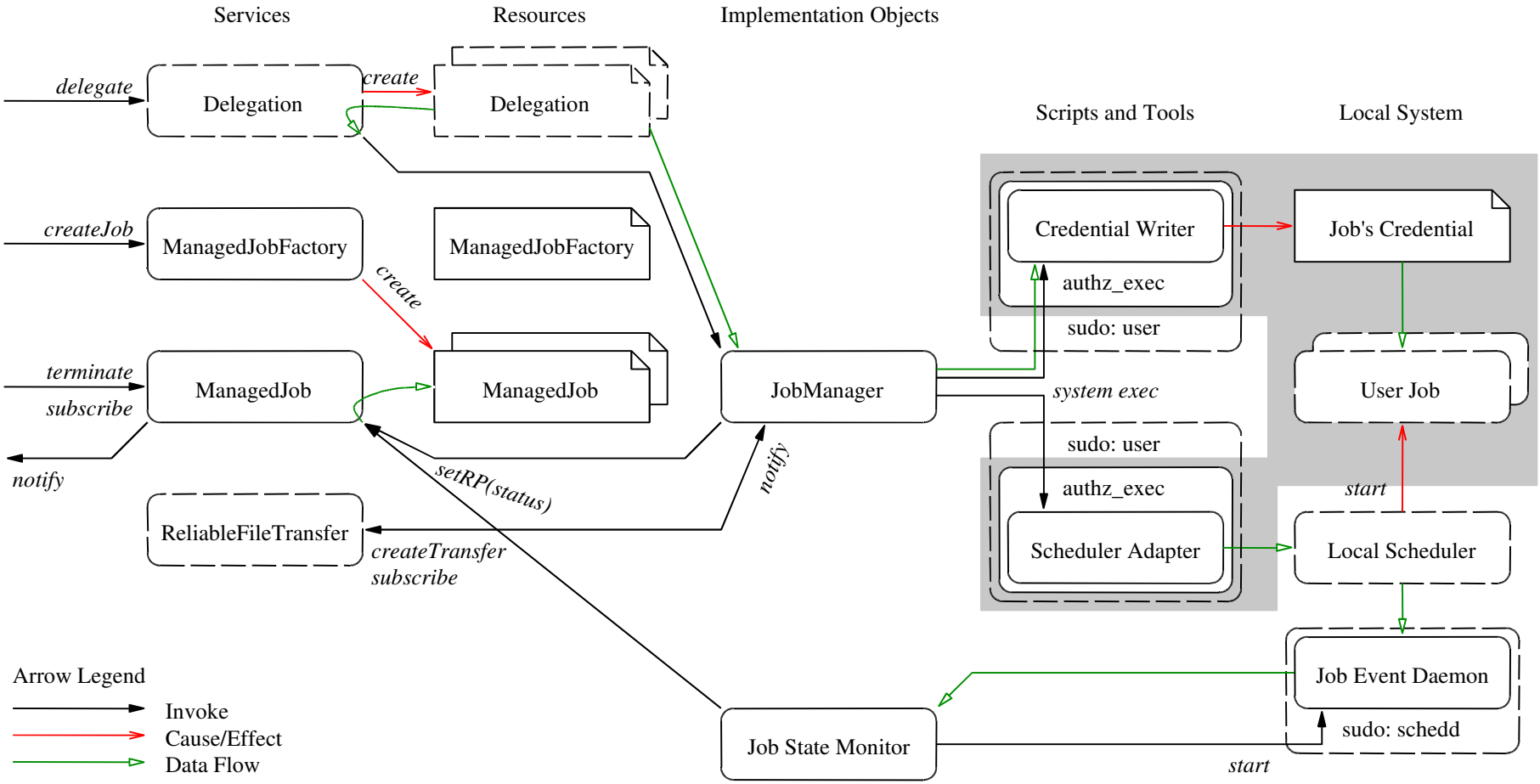
Possible GRAM Avenues

- 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
  - Ease migration

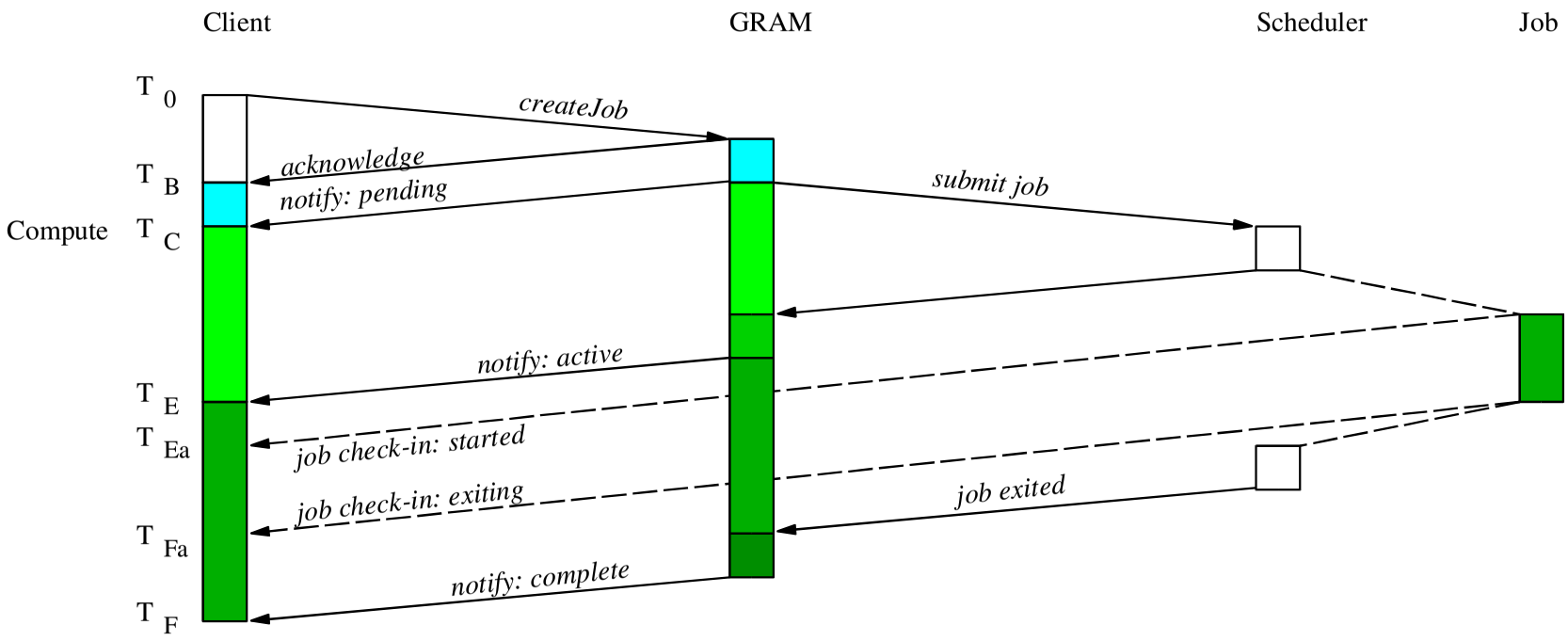
WS-GRAM Approach



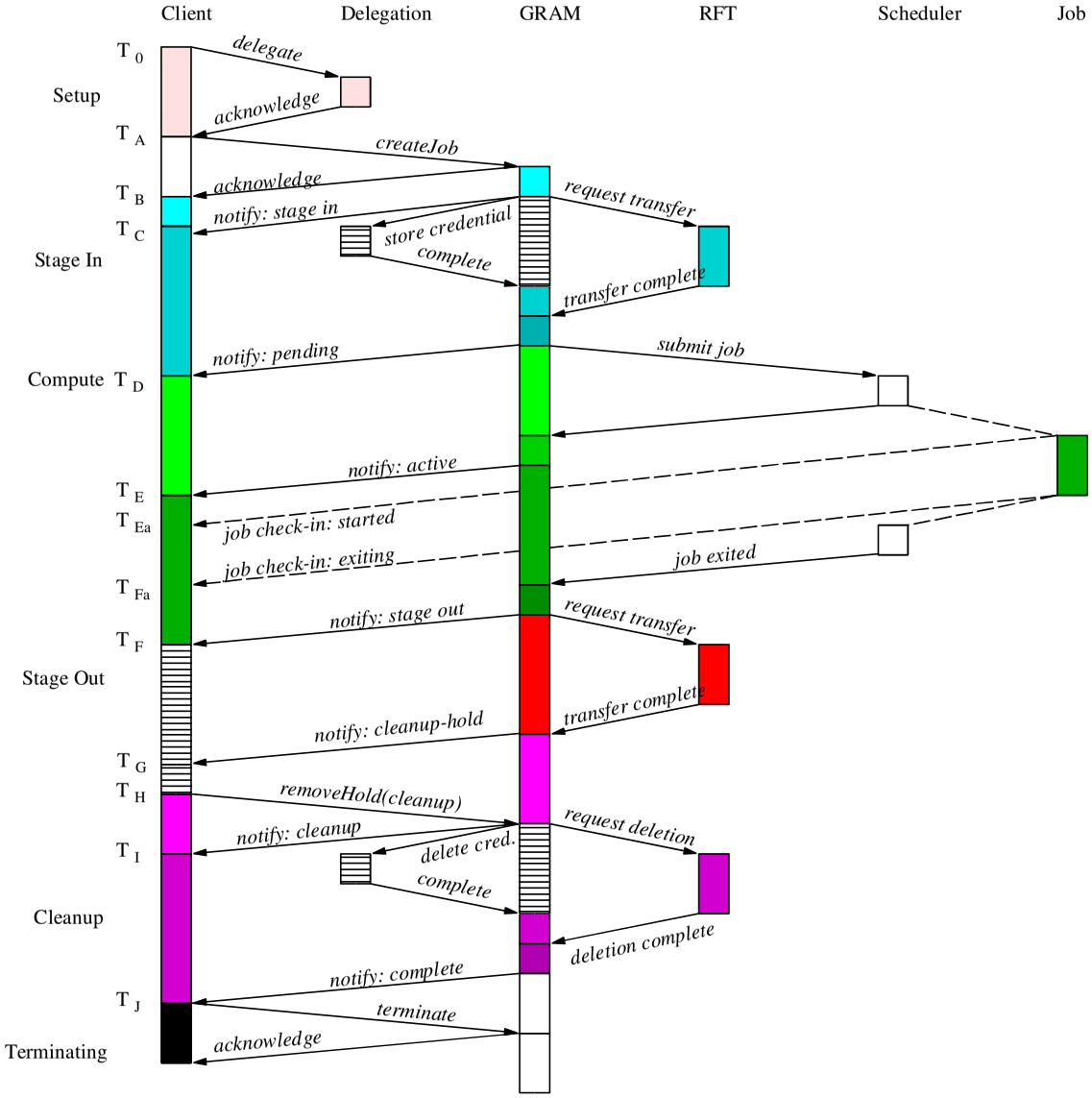
WS-GRAM Software Map



WS-GRAM Base Protocol



WS-GRAM Full Protocol



Positioning for Future

- Better modularity of job manager functions
  - Track improvements in composed services
  - 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?