



the globus project™
www.globus.org

Grid Scheduling Con Service-Level Agreement

Karl Czajkowski

The Globus Project

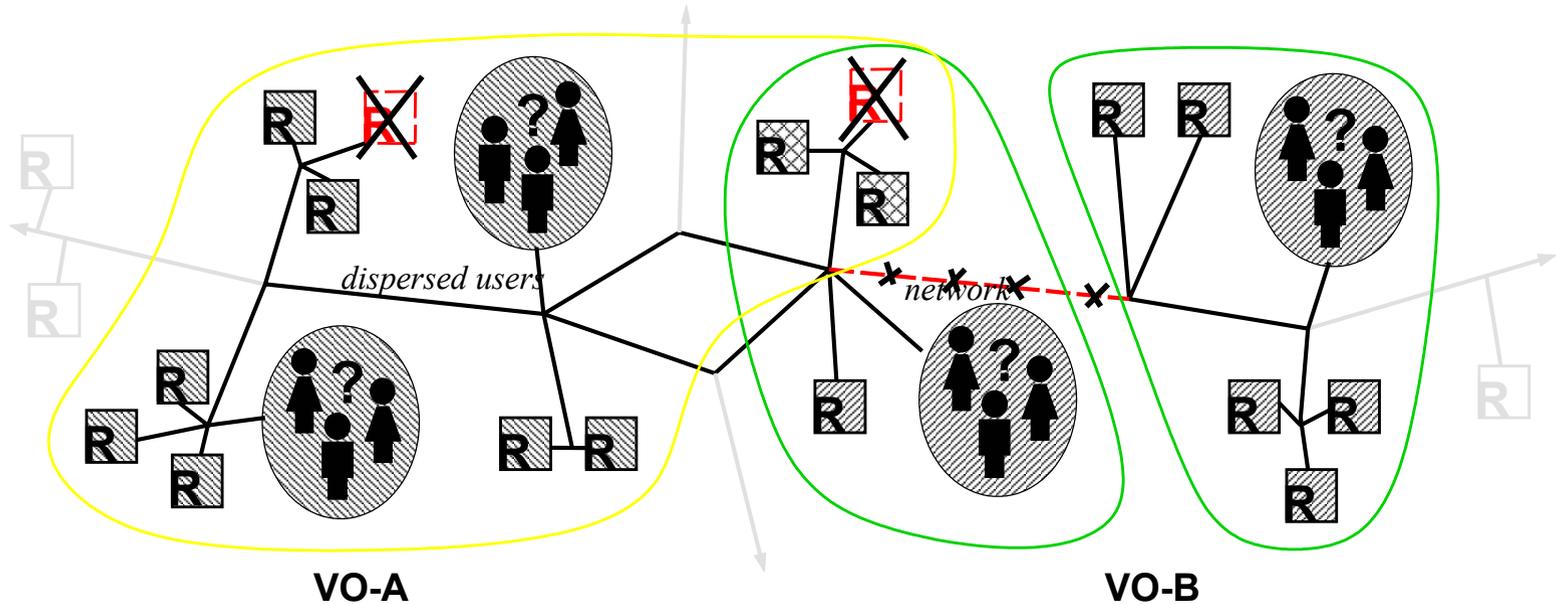
<http://www.globus.org/>



Sommario

- Introduzione ai Grid Environment
- Il Problema del Resource Management
 - Applicazioni in più domini
 - Obiettivi del proprietario vs. obiettivi delle applicazioni
- Una Architettura Aperta per Gestire Risorse
 - Service-Level Agreement (SLA)
 - GRAM e Servizi Gestiti
 - Sviluppi

Ambiente di Risorse di Griglia



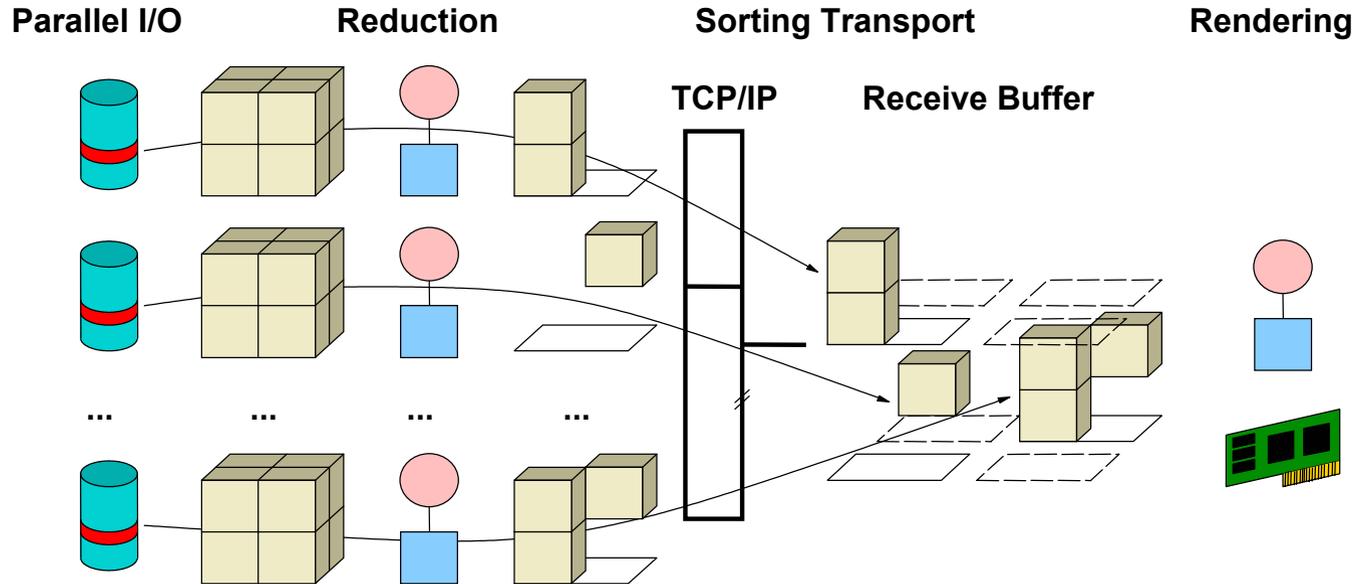
- Utenti e Risorse Distribuiti
- Stato delle Risorse Variabile
- Raggruppamento e Connettività Variabile
- Scheduling e Politiche Decentralizzate



Conflitti Sociali/di Politiche

- **Obiettivi dell'Applicazione**
 - Utenti: scadenze e obiettivi di disponibilità
 - Applicazioni: servono risorse coordinate
- **Obiettivi del proprietario della risorsa**
 - Politiche verso gli utenti
 - Obiettivi di ottimizzazione
- **Gli obiettivi della Comunità Emergono come:**
 - Un aggregato utente/applicazione?
 - Una risorsa virtuale? Entrambi!

Esempio Data-Intensive



- Requisiti di risorse Concorrenti
 - Memoria Elevata, elaborazione, rete, grafica
- Il percorso dei dati coinvolge domini autonomi

Iniziale Co-Allocazione in Griglie

- SF-Express (1997-8)
 - Simulazione real-time
 - 12+ supercomputer, 1400 processori
- Richiedeva advance reservation
 - Prenotazione per telefono!
 - Globus DUROC software per la sincronizzazione della fase di startup
 - Oltre 45 minuti per recuperare un fallimento
- Usato oggi in MPICH-G2 (MPI library)



Scheduling Tradizionale

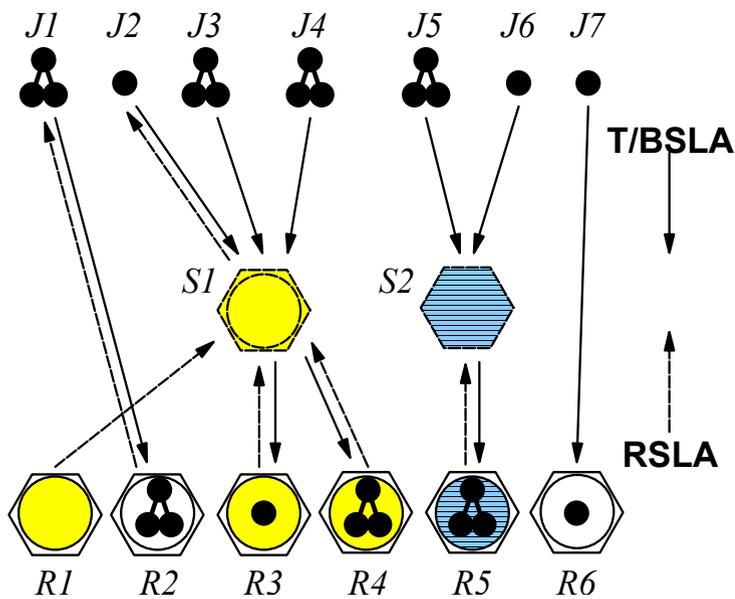
- **Modello Sistema Chiuso**
 - Assume un proprietario/autorità globale
 - Applicazioni “sandboxed” senza interazioni
 - “Lancia un job oltre la rete e aspetta”
- **Utilizzazione come Metrica Primaria**
 - Code batch profonde permettono un uso più efficiente
 - Nessun incentivo per soddisfare lo scheduling dell’utente
- **Politiche contro il sito**
 - Gli utenti usano trucchi per “farsi gioco” del sito.



Un Modello di Negoziazione Aperto

- Risorse in un Contesto Globale
 - Avviso e negoziazione
 - Interfaccia Client Remota normalizzata
 - Le risorse mantengono l'autonomia
- Utenti o Agenti *Collegano* le Risorse
 - Gestione della fornitura e sottomissione di task
 - Azioni Coordinate tra domini
- Mediazione Community-based
 - Coordinamento per interessi collettivi

Esempio di Community Scheduling



- **Utenti individuali**
 - Richiedono servizi
 - Hanno obiettivi applicativi
- **Community schedulers**
 - Servizi di brokering
 - Scheduling Aggregato
- **Risorse individuali**
 - Forniscono servizi
 - Hanno autonomia di politiche
 - Servono i clienti



Fasi di Negoziazione

- **Discovery**
 - “Quali sono le risorse di interesse?”
 - Trova i service providers
- **Monitoring**
 - “Cosa sta avvenendo con le risorse?”
 - Confronta i service providers
- ***Service-Level Agreement***
 - “Loro mi forniscono quello che serve a me?”
 - **Il Problema centrale del Resource Management**
 - Questo processo può iterare per trovare un adattamento.



Service-Level Agreement

- Tre tipi di SLA
 - Sottomissione di *Task* (fai qualcosa)
 - Prenotazione di una *Risorsa* (pre-accordo)
 - *Binding* di task/risorsa (chiedi di riservare)
- Protocollo semplice per negoziare SLA
 - Negoziazione semplice a due parti (2-party)
 - > Supporto per un modello di base offerta/accettazione
 - > Modelli opzionali di contro-offerta
 - > Fase di commitment variabile per promesse più vincolanti
 - Un Cliente può mantenere più 2-party SLA

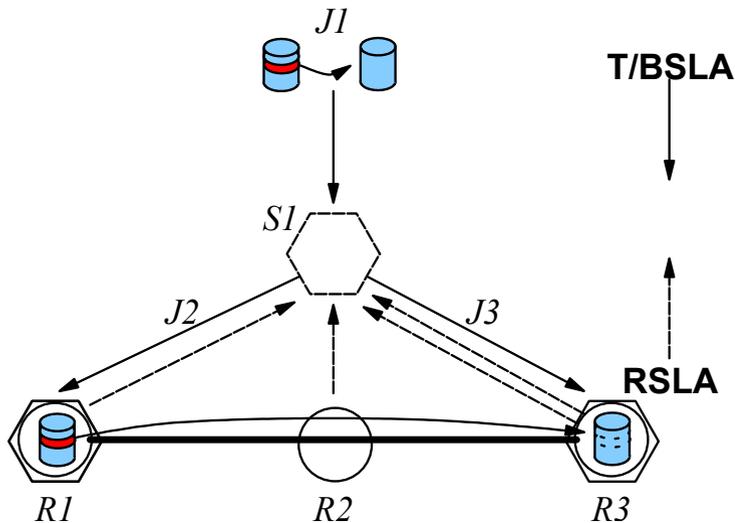


Molti Tipi di Servizi

- **Bisogna supportare eterogeneità di servizi**
 - Risorse
 - > Hardware: dischi, CPU, memoria, rete, display...
 - > Logiche: account, servizi, ...
 - > Capabilities: spazio, throughput, ...
 - Task
 - > Dati: files, read/write dati
 - > Elaborazione: esecuzione, job sospeso/swapped
- **SLA usano linguaggi con termini di dominio**
 - Isolare dettagli domain-specific



Estensione di Dominio : File Transfer



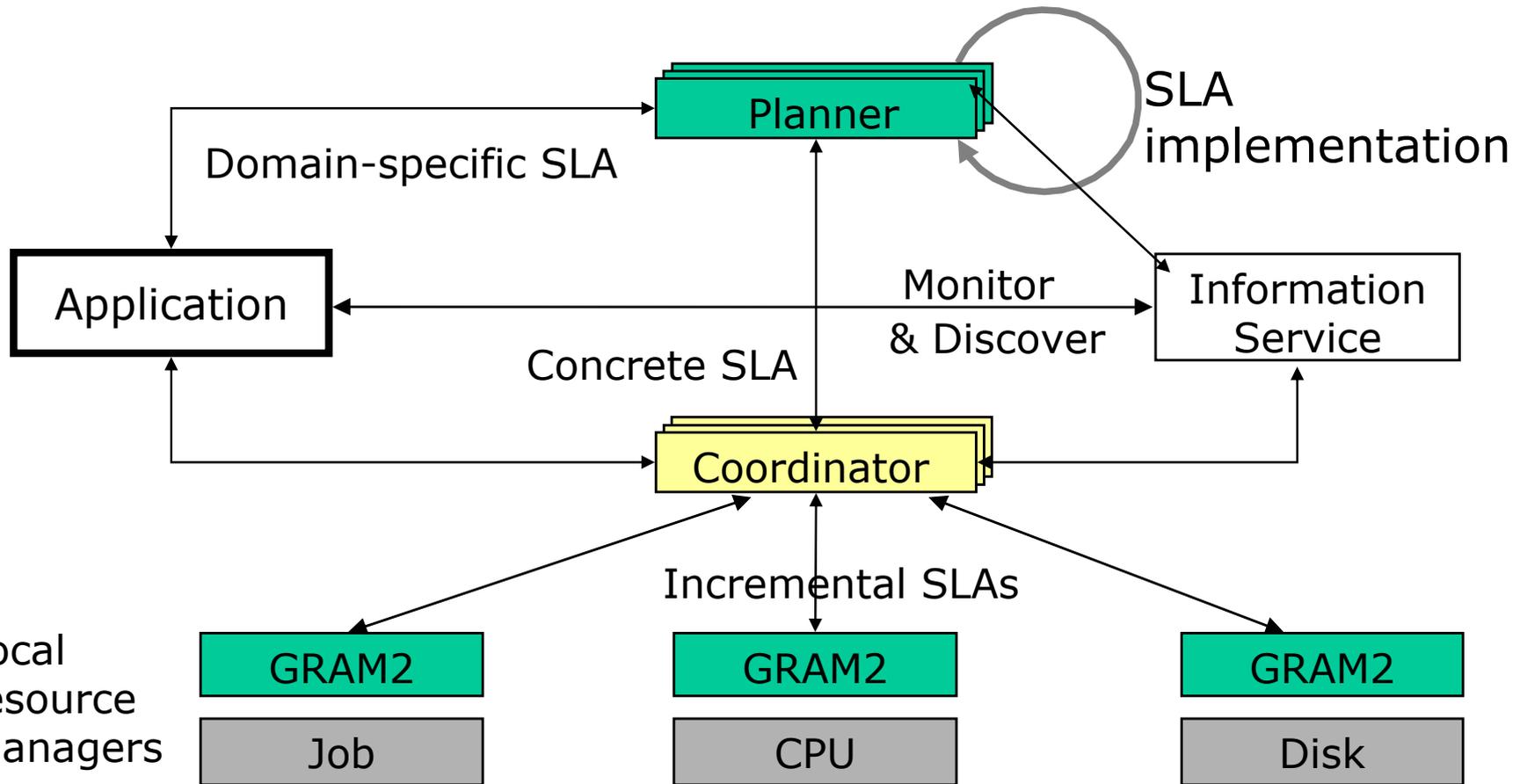
- Singolo obiettivo
 - Trasferimento con “deadline” sicura
- Scheduler Specializzato
 - Servizi semplici di brokering
 - Creazione di nuovi servizi
 - > Logica di gestione di guasti
- Risorse Distribuite
 - Spazio di memoria
 - Banda di accesso ai dati
 - Banda di rete

Sfide Tecniche

- Requisiti di Sicurezza Complessi
- Scalabilità Globale
 - Obiettivi simili ad Internet
 - Infrastruttura Interoperabile
 - Configurabile (su politiche) per ragioni sociali
- Permanenza o “Evolve in Place”
 - Non si può mettere tutti off-line per motivi di servizio
 - Nel tempo: upgrade, estendi, adatta
 - Accetta l’eterogeneità



Architettura GRAM



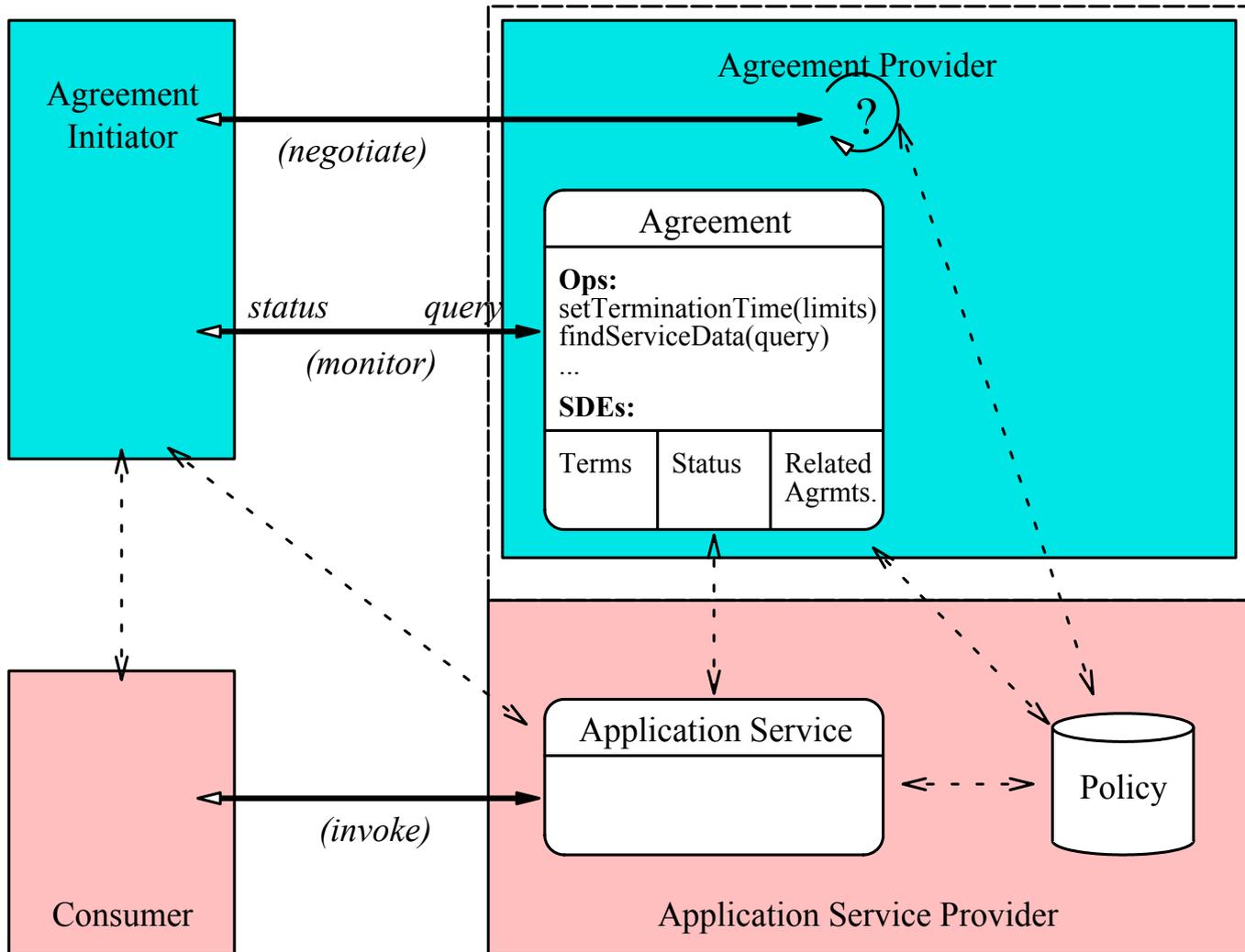


WS-Agreement

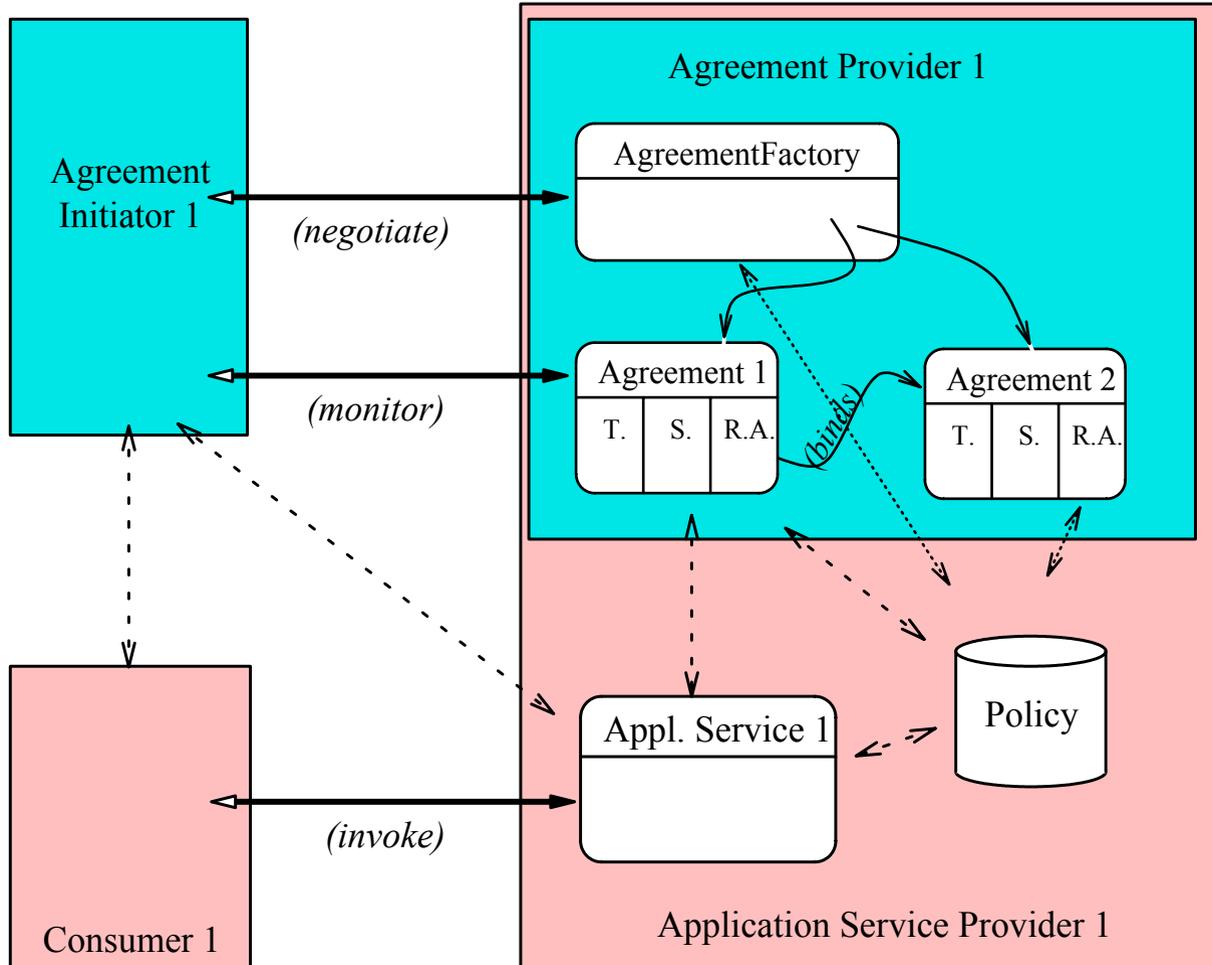
- Nuovo impegno di standardizzazione
- Generalizza le idee del GRAM
 - Architettura Service-oriented
 - Una *Risorsa* diventa un *Service Provider*
 - I Task diventano *Negotiated Services*
 - I SLA presentati come *Agreement Services*
- Mantiene il supporto all'estensione con termini di dominio.



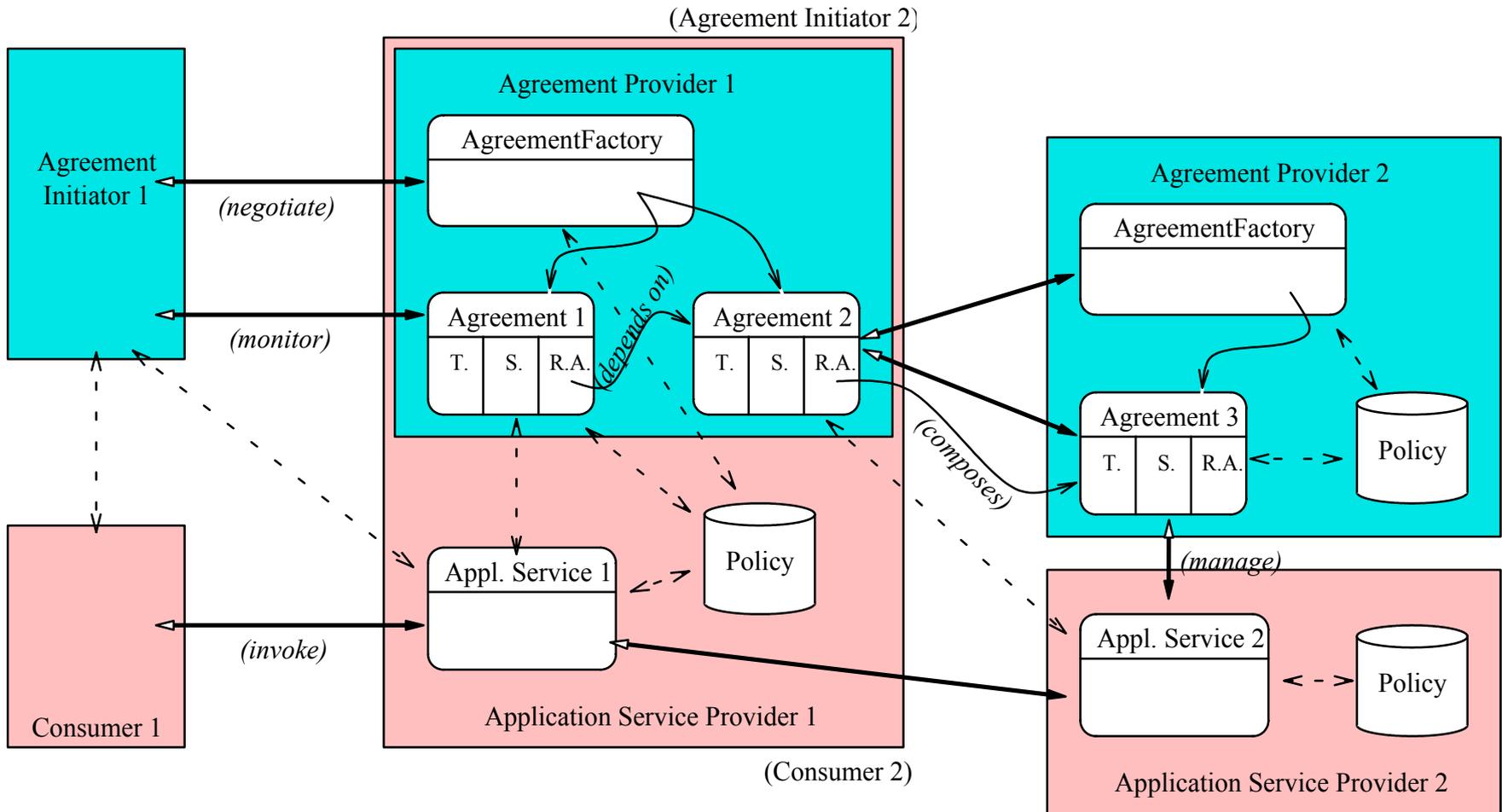
Entità in un WS-Agreement



WS-Agreement richiede una Gestione



Virtualized Providers





Jobs Basati su Agreement

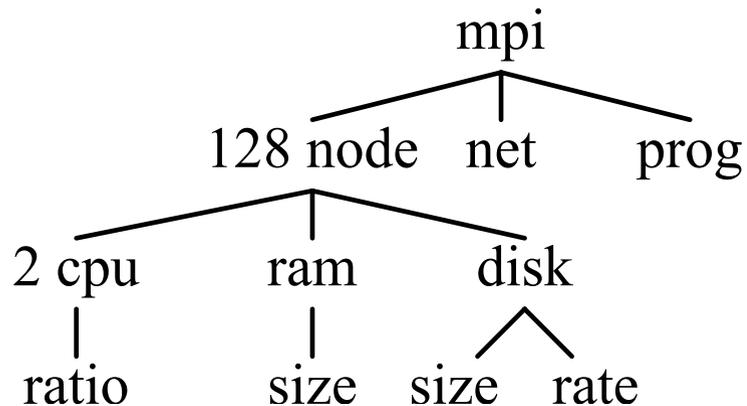
- Agreement rappresenta una “queue entry”
 - Commitment con parametri dei job etc.
- Agreement Provider
 - Sistema di Job scheduler/Queuing
 - Interfaccia di gestione verso il service provider
- Service Provider
 - Risorsa allocata (nodi di elaborazione)
- Il servizio è l'esecuzione di Job



Prenotazione Anticipata di Job

- Commitment del servizio basato sullo Scheduling
 - Richiede termini SLA basati sullo schedule
- Pre-Agreement Opzionale (RSLA)
 - Agreement per facilitare futuri Job Agreement
 - Caratterizza le risorse virtuali necessarie al Job
 - Potrebbe uno necessitare di termini orientati al job
- Job Agreement quasi normale
 - Può sfruttare il Pre-Agreement
 - > Fa riferimento a promesse esistenti di scheduling di risorse
 - Può ottenere il commitment in un colpo solo
 - > Include direttamente termini di schedule
 - > (Si può pensare come ad una richiesta atomica anticipata)

Serve una Descrizione Complessa



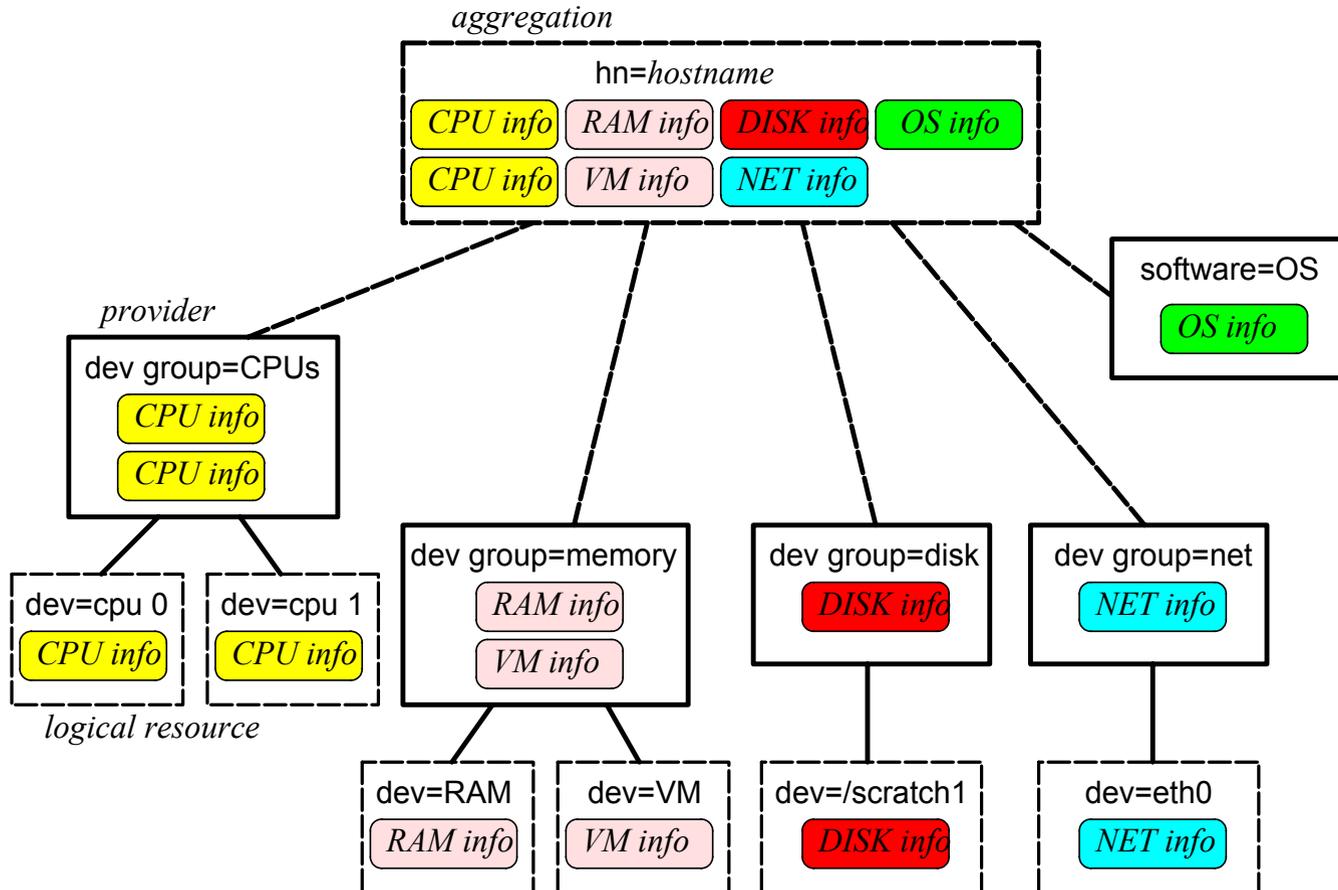
- 128 nodi fisici
 - Topologia fisica
 - > Interconnessione
 - > RAM, dimens. del disco
 - Soggetto a Resource SLA
- Job MPI Singolo
 - Soggetto a Task SLA
 - Può far riferimento a RSLAs

Requisiti di Qualità

- Parametri Real-time
 - > CPU, performance del disco
- Soggetto a Binding SLA



Modelli di Risorse MDS (Storia)





Modelli Futuri

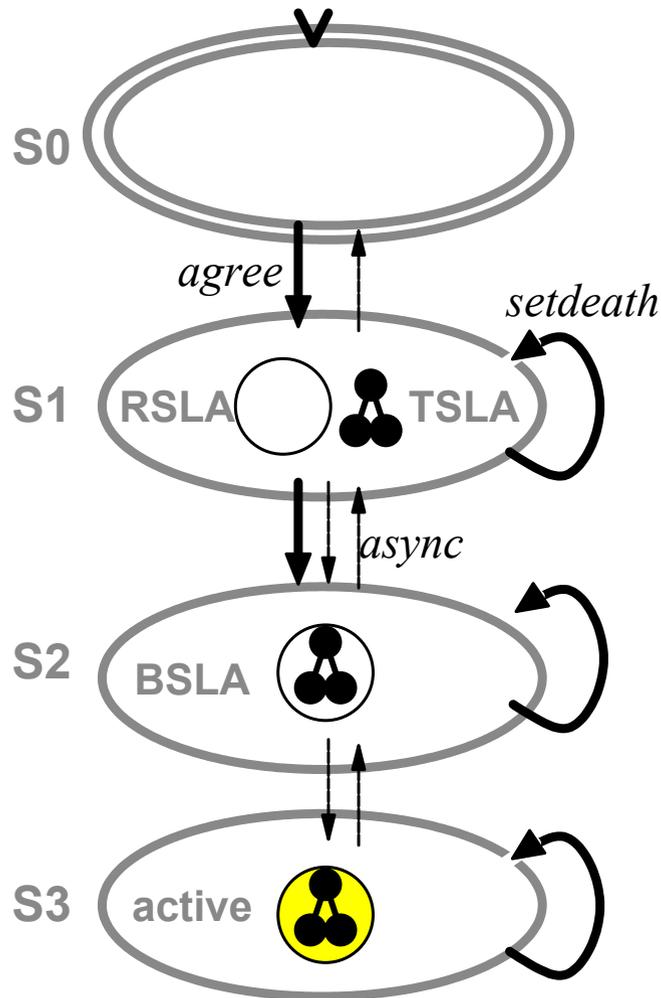
- **Descrizione del comportamento dei servizi**
 - Modello unificato dei termini dei servizi
 - Cattura i requisiti di utenti/applicazioni
 - Cattura le capacità dei provider
- **Core meta-language**
 - Facilita la progettazione di pianificazioni/decisioni
 - Estende con concetti di dominio
 - Mark-up di negoziabilità estensibile
 - > Cattura intervalli di negoziabilità per termini variabili
 - > Cattura l'importanza di termini (richiesto/opzionale)
 - > Cattura i costi delle opzioni (tasse/penalità)



Tipi di SLA in Dettaglio

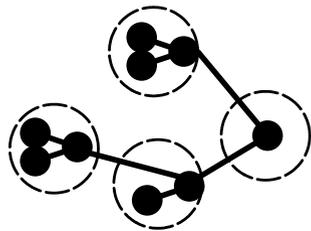
- **Resource SLA (RSLA), cioè *prenotazione***
 - Una promessa di disponibilità di risorse
 - > Un Cliente deve utilizzare la promessa in successivi SLA
- **Task SLA (TSLA), cioè *esecuzione***
 - Una promessa di eseguire un task
 - > Task requirements complessi
 - > Può far riferimento ad un RSLA (binding implicito)
- **Binding SLA (BSLA), cioè *richiesta***
 - Collega una risorsa potenziale ad un TSLA
 - > Può far riferimento ad un RSLA (o lo ottiene implicitamente)
 - > Può essere creato in previsione di un task

Ciclo di Vita di una Risorsa

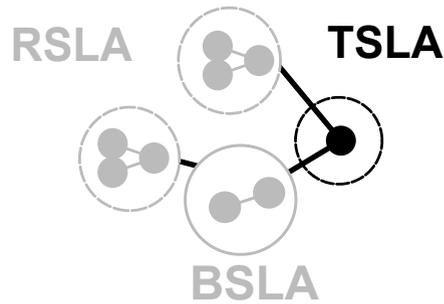


- **S0:** Parte senza SLA
- **S1:** Crea dei SLA
 - TSLA o RSLA
- **S2:** Collega task/risorsa
 - BSLA esplicito
 - Schedule del provider implicito
- **S3:** Attiva un task
 - Consumo della risorsa
- **Ritorna ad S0**
 - Al completamento del task
 - Per fine del servizio
 - Per fallimento

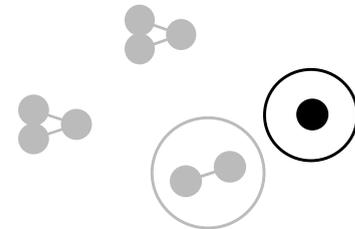
Negoziazione Incrementale



User goal



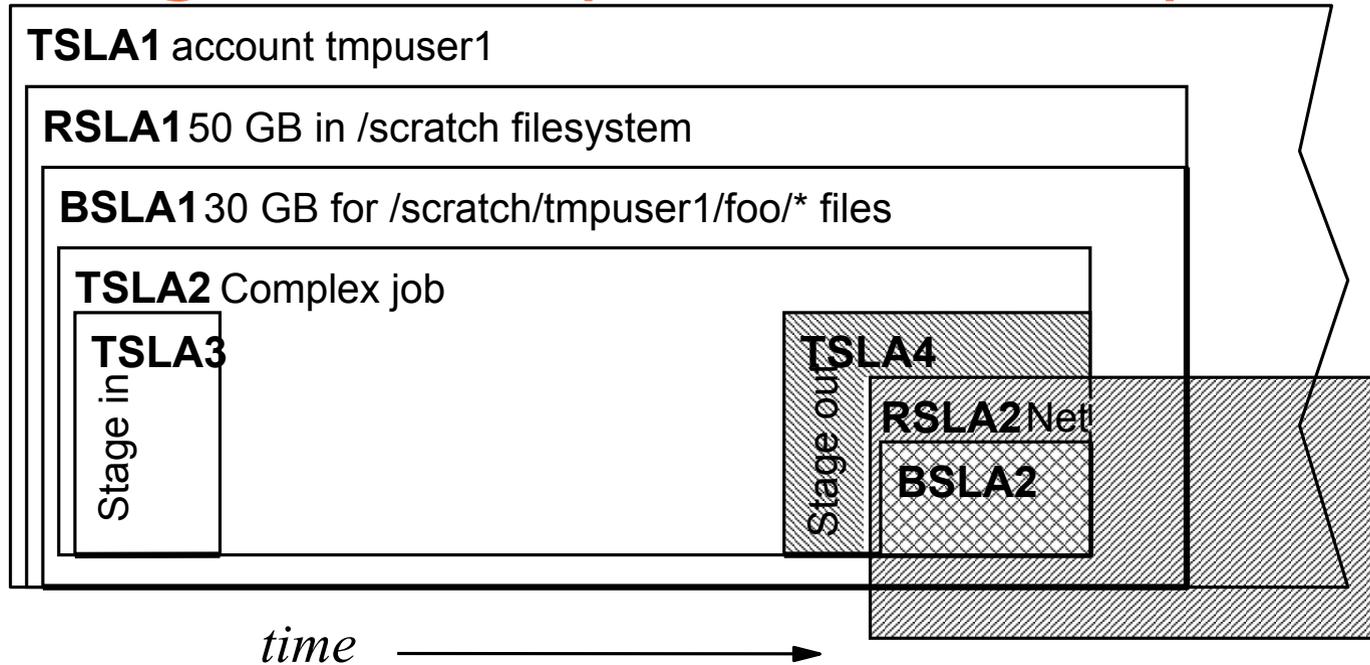
SLAs



Resource state

- RSLA: riserva le risorse per usi futuri
- TSLA: sottomette task allo scheduler
- BSLA: collega una prenotazione ad un task
- Le risorse cambiano stato a causa dei SLA e alle decisioni dello scheduler

Collegare SLA per Casi Complessi



- SLA dipendenti si innestano intrinsecamente
 - BSLA2 è definito in termini di RSLA2 e TSLA4
- SLA "incatenati" semplificano la negoziazione



Related Work

- Proposte di ricerca
 - Condor Matchmaking
 - Economy-based Scheduling
 - Work-flow Planning
- Esempi di Scheduler Commerciali
 - Molti esempi per siti tradizionali
 - Platform Computing
 - > LSF usato per *molti job*
 - > MultiCluster per condivisione di risorse tra siti
 - IBM eWLM
 - > Goal-based provisioning di flussi transazionali



Condor Matchmaking

- Essenzialmente: un algoritmo di scheduling
- Euristiche per accoppiare job e risorse
 - Associa annunci simmetrici
 - Ottimo per matching di grandi dimensioni
- Offre una vista a “sistema chiuso”
 - Usa le risorse attraverso il “prestito”
 - Ambiente job sandboxed
 - Favorisce l’integrazione verticale rispetto alla generalità
 - High-throughput system

Condor sul GRAM

- Condor già usa il GRAM
 1. Il GRAM tratta Condor come uno scheduler locale
 2. Condor usa il GRAM per accedere alle risorse
- Condor maps to SLA architecture
 - Pubblica ClassAd di risorse
 - Sottomette job ClassAd (come TSLA)
 - Matchmaker è un Community Scheduler
 - Necessita di scalabilità SLA per essere di utilità pratica



Lavoro Futuro

- **Interazione SLA - politiche**
 - Negoziazione SLA soggetta a politiche
 - > Una SLA influenza un'altra, es. RSLA subdivision
 - > Un cliente "più importante" di altri
 - SLA implementato da politiche di basso livello
 - > Domain-specific SLA associate a resource SLA
 - > Resource SLA associate a meccanismi di controllo
- **Caratterizzazione di risorse**
 - Pubblicazione di risorse: opzioni, costi
 - Linguaggio di capability interoperabile

Conclusioni

- **Gestione di SLA Generici**
 - Compositazionale per scenari complessi
 - Estensibile per requisiti particolari
 - Richiede ulteriore lavoro per i Grid service
 - > Per descrivere job, resource requirements, ecc.
- **Evoluzione del RM nel Globus Toolkit**
 - GRAM in evoluzione dal 1997
 - WS-Agreement standard in sviluppo