ISSGC
The 4th International
Summer School on Grid Computing

Session 2

Monday 10th July

Malcolm Atkinson

Distributed Systems: Introduction, Principles & Foundations

- ***Issues you can't avoid***
  - Lack of Complete Knowledge (LOCK)
  - Latency
  - Heterogeneity
  - Autonomy
  - Unreliability
  - Change

This is NOT easy

- ***A Challenging goal***
  - balance technical feasibility
  - against virtual homogeneity, stability and reliability
- ***Appropriate balance between usability and productivity***
  - while remaining affordable, manageable and maintainable

Lack of Complete Knowledge

- Technical origins of LoCK
  - Dynamics of systems involve very large state spaces
    - Can’t track or explore or all the states
  - Latency prevents up-to-date knowledge being available
    - By the time a notification of a state change arrives the state may have changed again
  - Failures inhibit information propagation
  - Unanticipated failure modes

Lack of Complete Knowledge 2

- Human origins of LoCK
  - lack of understanding
  - Incomplete simplified models
  - Intractable models
  - Poor & incomplete descriptions
  - Erroneous descriptions
- Socio-Economic effects generate LoCK
  - Autonomous owners do not choose to reveal all
    - About their services, resources and performance
  - Intermediaries aggregate & simplify

LoCK Counter Strategies

- Improve the quality of the available knowledge
  - Better static information
  - Better information collection & dissemination
- Improve quality of the Distributed System Models
  - Prove invariants that algorithms can exploit
  - Test axioms with real systems
- Build algorithms that behave reasonably well
  - When they have incomplete knowledge

Latency

- It is always going to be there
  - Consequence of signal transmission times
  - Consequence of messages / packets in queues
  - Consequence of message processing time
  - Errors cause retries
- It gets worse
  - Geographic scale increases latency
  - System complexity increases number of queues
  - System scale & complexity increase processing time
- Think about
  - How many operations a system can do while a message it sent reaches its destination, a reply is formed and the reply travels back

Latency Counter Strategies

- Design algorithms that require fewer round trips
  - This is THE complexity measure!
  - Batching requests and responses
- Shorten distance to get information
  - Caching
    - But may be stale data!
  - Move data to computation
    - But be smart about which data when
  - Move computation to data
    - Succinct computation & volumes of data
    - But safety and privacy issues arise

Heterogeneity

- Hardware variation
  - Different computer architectures
    - Big endians v little endians
    - Number representation
    - Address length
    - Performance
  - Different Storage systems
    - Architectures
    - Technologies
    - Available operations
  - Different Instrument systems
    - Accepting different control inputs
    - Generating different output data streams

Some of the variation is wanted and exploited

Heterogeneity 2

- Operating System variation
  - Different O/S architectures
    - Unix families & versions
    - Windows families and versions
    - Specialised O/S,
      e.g. for Instruments & Mobile devices
- Implementation system variation
  - Programming languages
  - Scripting languages
  - Workflow systems
  - Data models
  - Description languages
  - Many implementations of same functionality

Some of the variation is just makes work

Heterogeneity Counter Measures

- Invest in *virtual* Homogeneity
  - Agree standards (formally or *de facto*)
  - Introduce intermediate code
    - That hides unwanted variation
    - Presenting it in standard form
  - But this has high cost
    - Developing the standard
    - Developing the intermediate code
    - Executing the intermediate code
  - It may hide variations some want
    - Provide direct access to facilities as well
    - But this may inhibit optimisation & automation

May be work in progress

Heterogeneity Counter Measures 2

- Automatically manage diversity
  - Manual agreement and construction of virtual homogeneity will not scale & compose
  - Develop abstract and higher level models
  - Describe each component
  - Generate the adaptations as needed from these descriptions
- Not yet achievable for the general & complete systems
  - Relevant for specific domains

Autonomy and Change

- Necessary
  - To persuade organisations & individuals to engage
    - They need to control their own facilities
    - They have best knowledge to develop their services
    - Their business opportunity
  - Because coordinated change is unachievable
    - Systems & workloads are busy
    - Service commitments must be met
    - Large-scale scheduling of work is very hard
  - To correct errors
  - To plug vulnerabilities

Autonomy and Change 2

- What changes – local decisions
  - The underlying technology delivering a service
  - The operations available from a service
  - The semantics of the operations
  - Policy changes, e.g. authorisation rules, costs, …
- What changes – corporate decisions
  - Some agreed standard is changed
    - E.g. a new version of a protocol is introduced

Autonomy and change Counter Measures

- Users & other providers expect stability
- Agree some standards that are rarely changed
  - As a platform & framework
  - As a means of communicating change
- Introduce change-absorbing technology
  - Mark the protocols and services with version information
  - Transform between protocols when changes occur
  - Anneal the change out of the system
- Develop algorithms tolerant to change
  - Revalidate dependencies where they may change
  - Handle failures due to change

Unreliability

- Failures are inevitable
  - Equipment, software & operations errors
  - Network outages, Power outages, …
- Their effects must be localised
  - Cannot afford total system outages
- This is not easy
  - Each error may occur when system is in any state
  - The system is an unknown composition of subsystems
  - Errors often occur while other errors are still active
  - Errors often occur during error recovery actions
  - Errors may be caused by deliberate attack
  - Attackers may continue their attack

Unreliability Counter Measures

- Requires much R&D
- Continuous arms race as scale of Grids grow
- Ideal of a continuously available stable service
  - Not achievable – recognise that drops in response and local failures must be dealt with
- Design resilient architectures
- Design resilient algorithms
- Improve reliability of each component
- Distribute the responsibility
  - For failure detection
  - For recovery action

Service Oriented Architectures

Three Components

Registries

Register an available service
Send name & description

Service Consumers

Services

Three Components

Registries

Request a service
Send a description

Service Consumers

Services

Three Components

Registries

Service Consumers —Request service operation→ **Services**

Three Components

Registries

Service Consumers ← **Services**

Return result or Error

Composed behaviour

- Services are themselves consumers
  - They may compose and wrap other services
- The registry is itself a consumer
- A federation of registries may deal with registry services reliability & performance
- Observer services may report on quality of services and help with diagnostics
- Agreements between services may be set up
  - Service-Level Agreements
  - Permitting sustained interaction

Composed behaviour

- Services are themselves consumers
  - They may compose and wrap other services
- The registry is itself a container
- A federation of registries may deal with registry service reliability & performance
- Observer services may report on quality of service and help with diagnostics
- Agreements between services may be set up
  - Service-Level Agreements
  - Permitting sustained interaction

Requires Organising as an Architecture

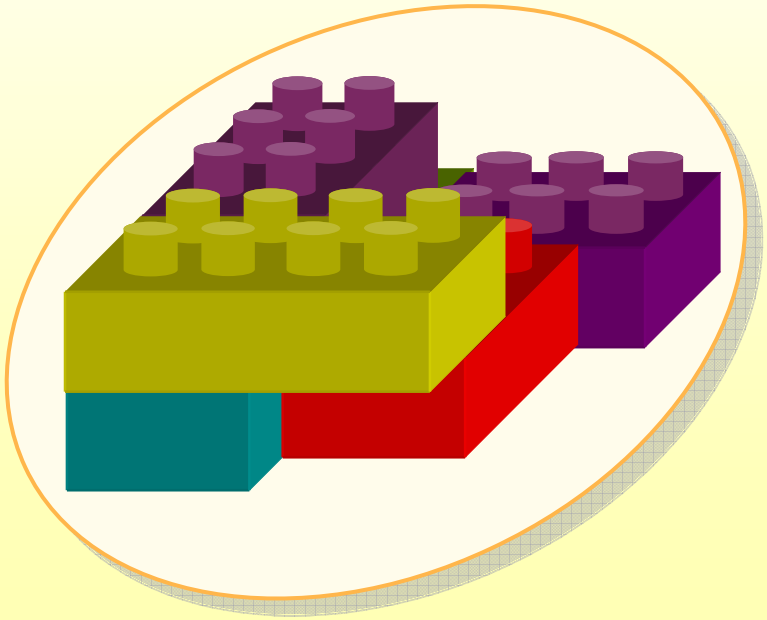
GGF Open Grid Services Architecture

The Open Grid Services Architecture

- An open, service-oriented architecture (SOA)
  - Resources as first-class entities
  - Dynamic service/resource creation and destruction
- Built on a Web services infrastructure
- Resource virtualization at the core
- Build grids from small number of standards-based components
  - Replaceable, coarse-grained
  - e.g. brokers
- Customizable
  - Support for dynamic, domain-specific content…
  - …within the same standardized framework

Why Use an SOA?

- Logical view of capabilities
- Relatively coarse-grained functions
- Reusable and composable behaviors
- Encapsulation of complex operations
- Naturally extendable framework
- Platform-neutral
  - machine *and* OS



SOA & Web Services: Key Benefits

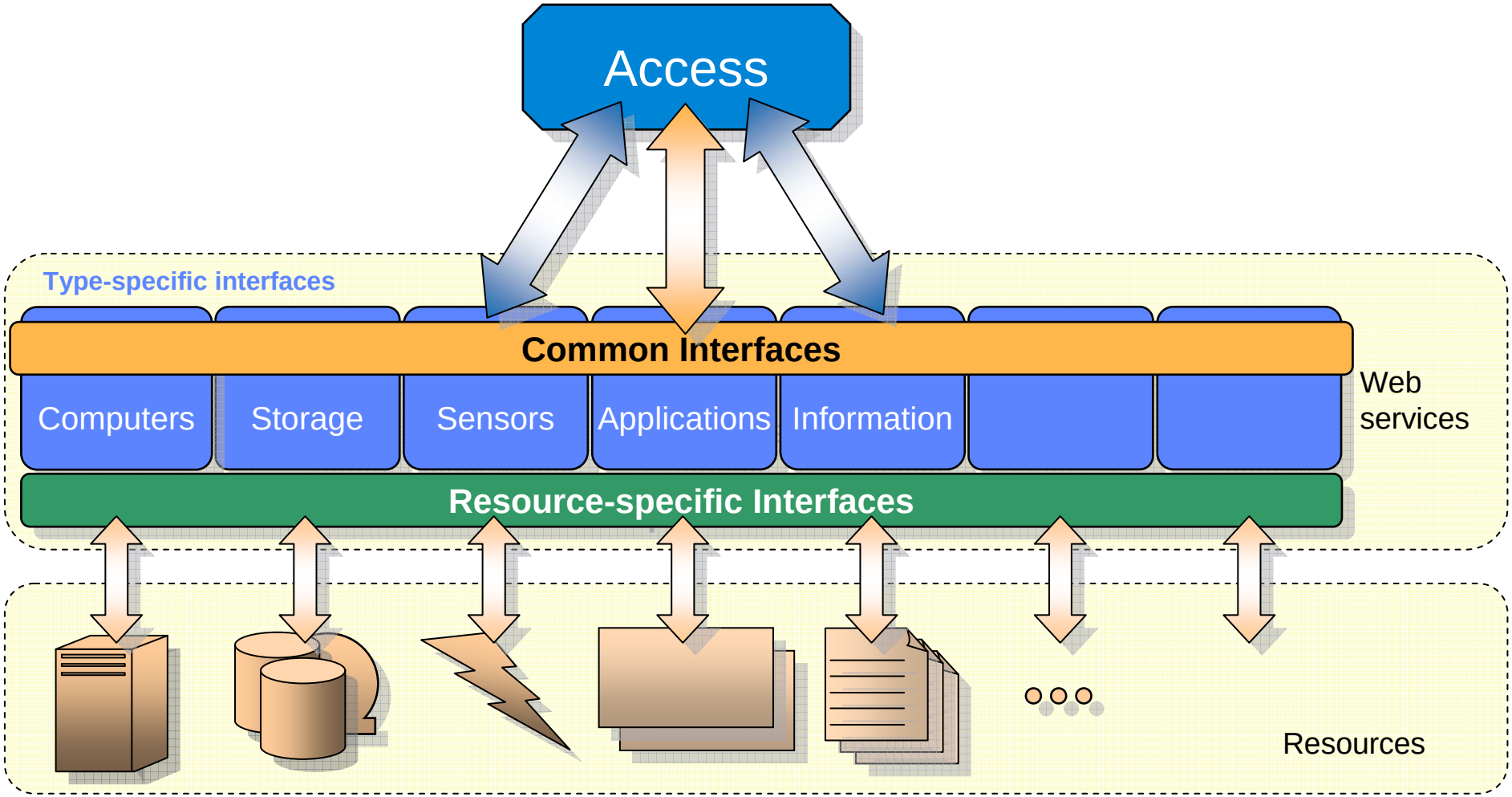
SOA

- Flexible
  - Locate services on any server
  - Relocate as necessary
  - Prospective clients find services using registries
- Scalable
  - Add & remove services as demand varies
- Replaceable
  - Update implementations without disruption to users
- Fault-tolerant
  - On failure, clients query registry for alternate services

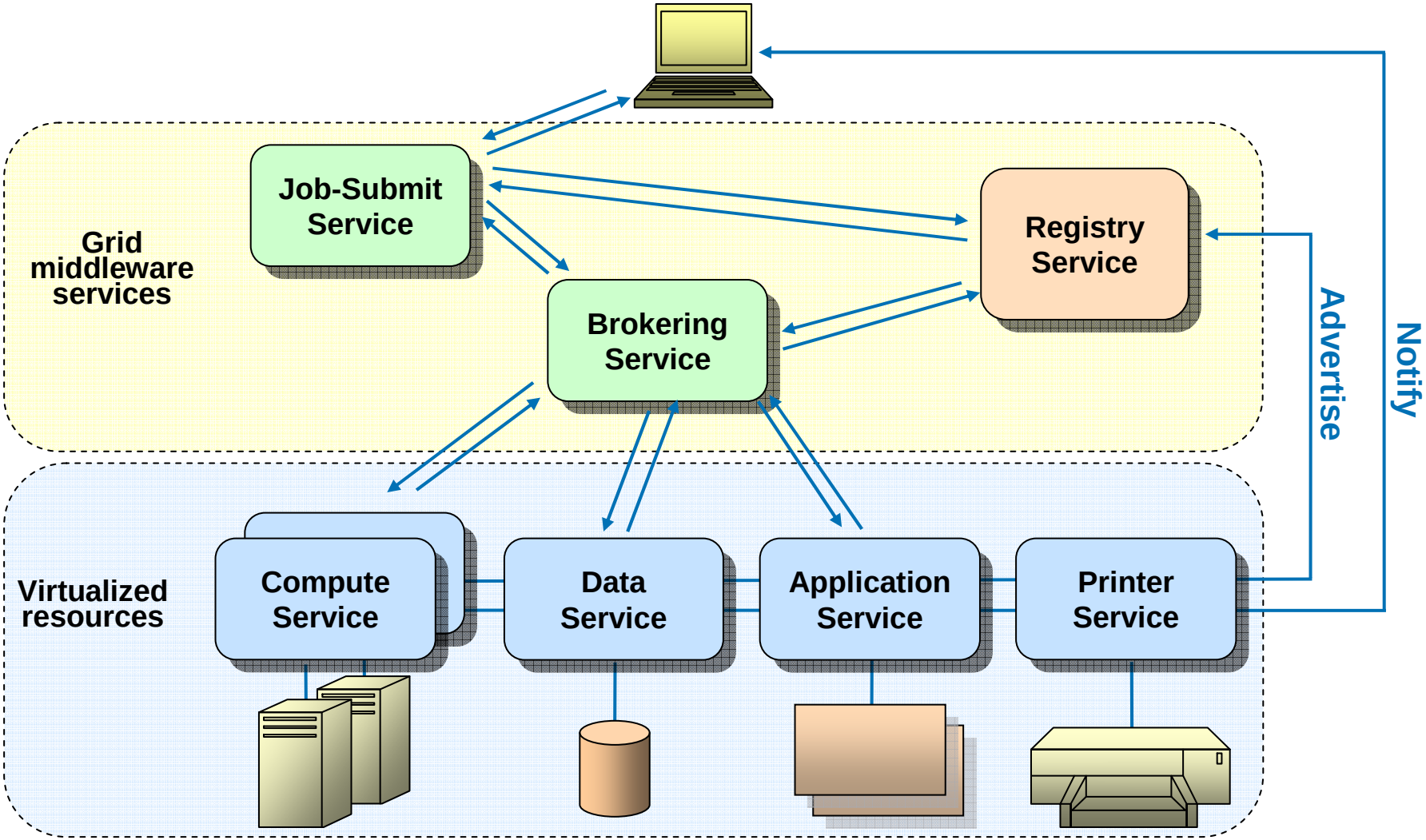
Web Services

- Interoperable
  - Growing number of industry standards
- Strong industry support
- Reduce time-to-value
  - Harness robust development tools for Web services
  - Decrease learning & implementation time
- Embrace and extend
  - Leverage effort in developing and driving consensus on standards
  - Focus limited resources on augmenting & adding standards as needed

Virtualizing Resources

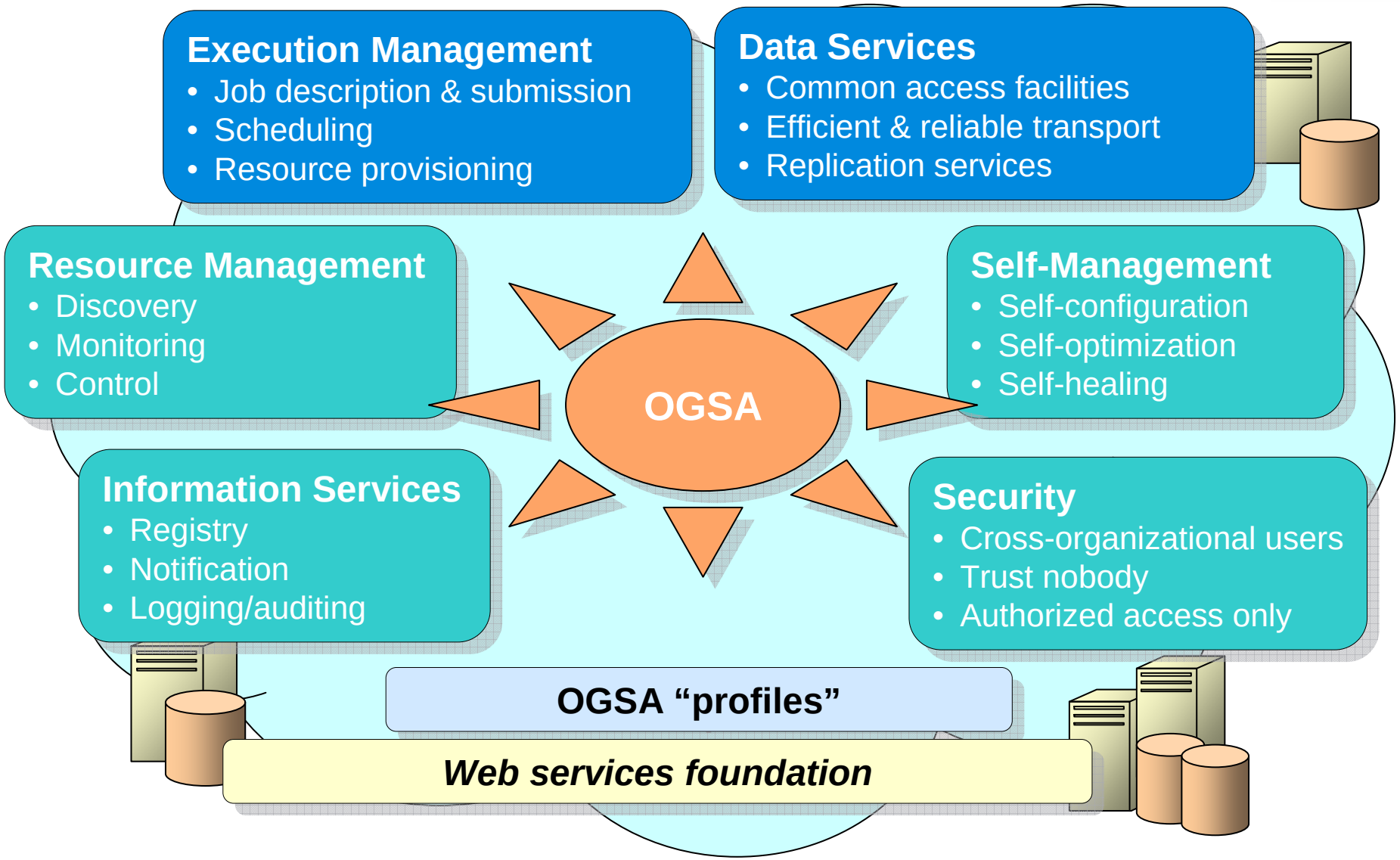


A Service-Oriented Grid



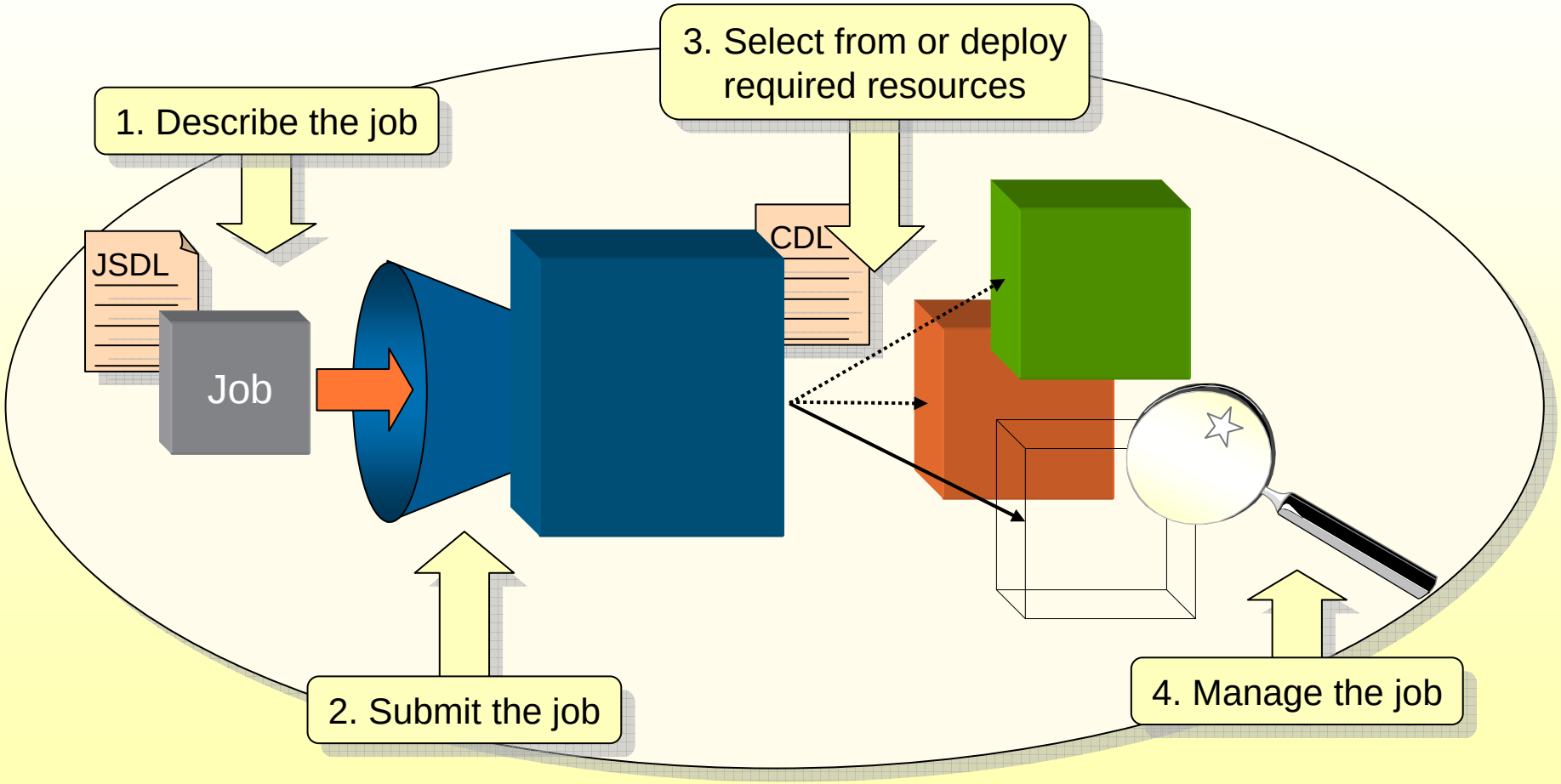
A Closer Look at OGSA

OGSA Capabilities



Execution Management

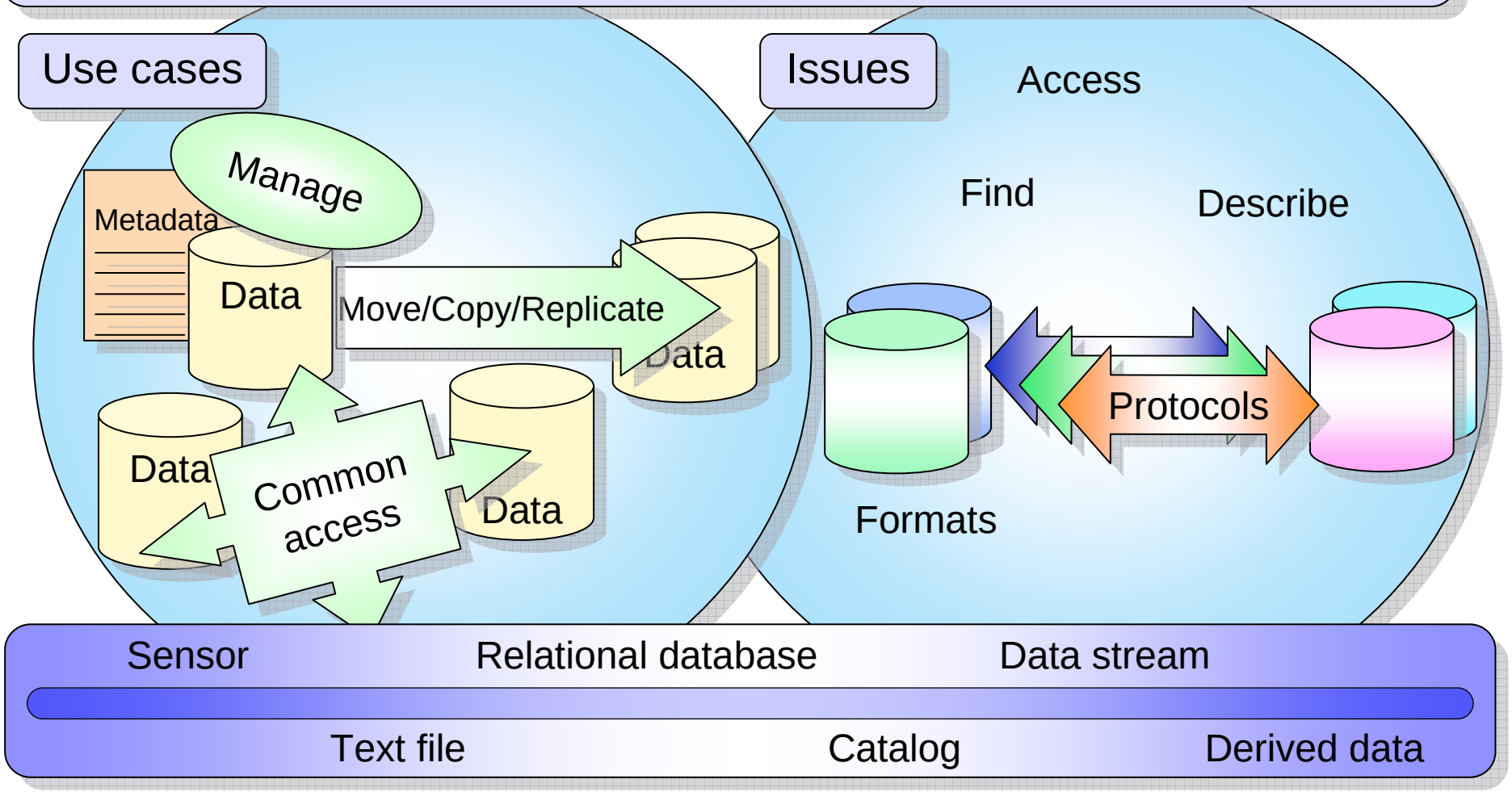
- The basic problem
  - Execute and manage jobs/services in the grid
  - Select from or provision required resources



Data Services

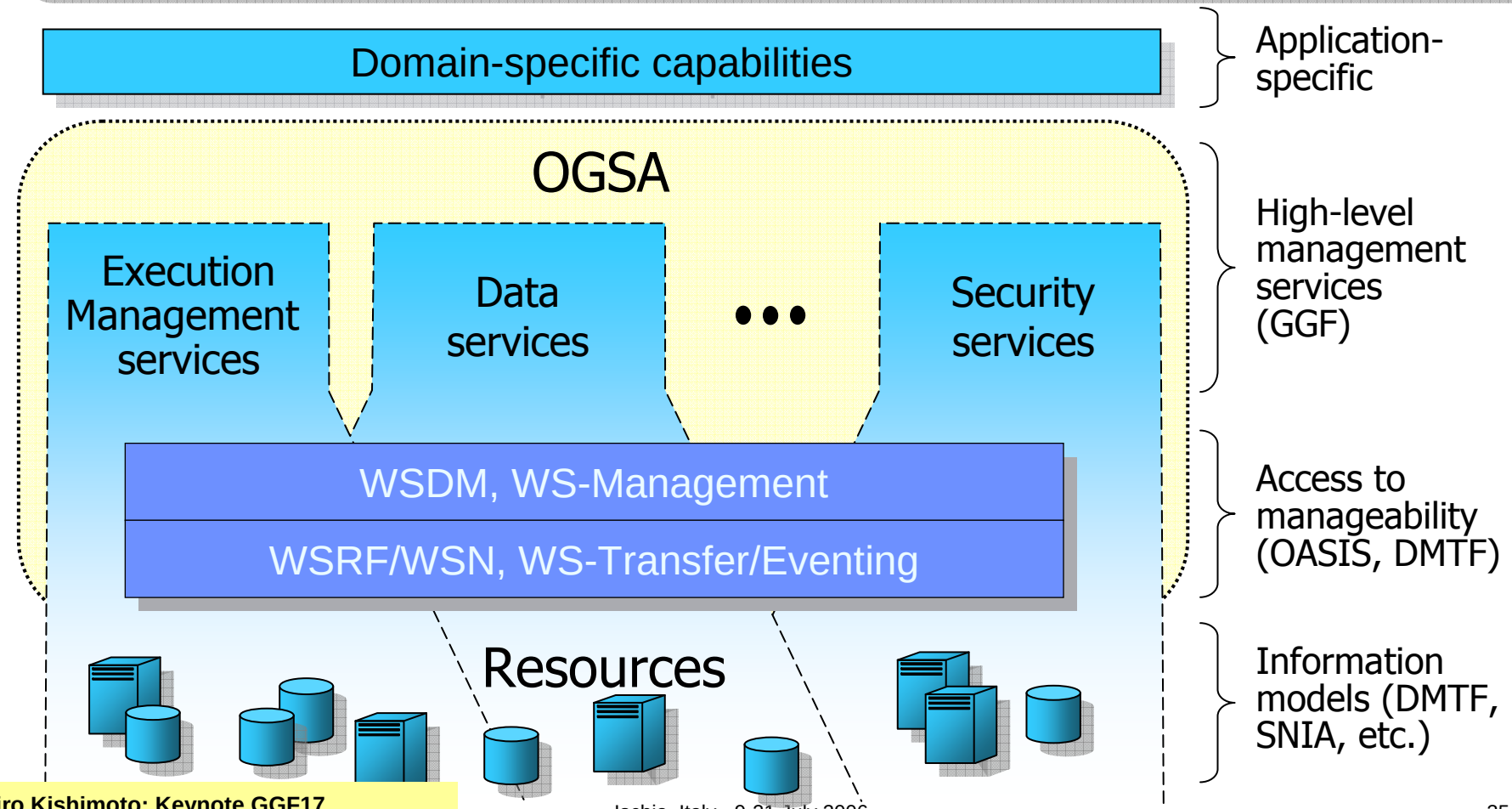
The basic problem

- Manage, transfer and access distributed data services and resources



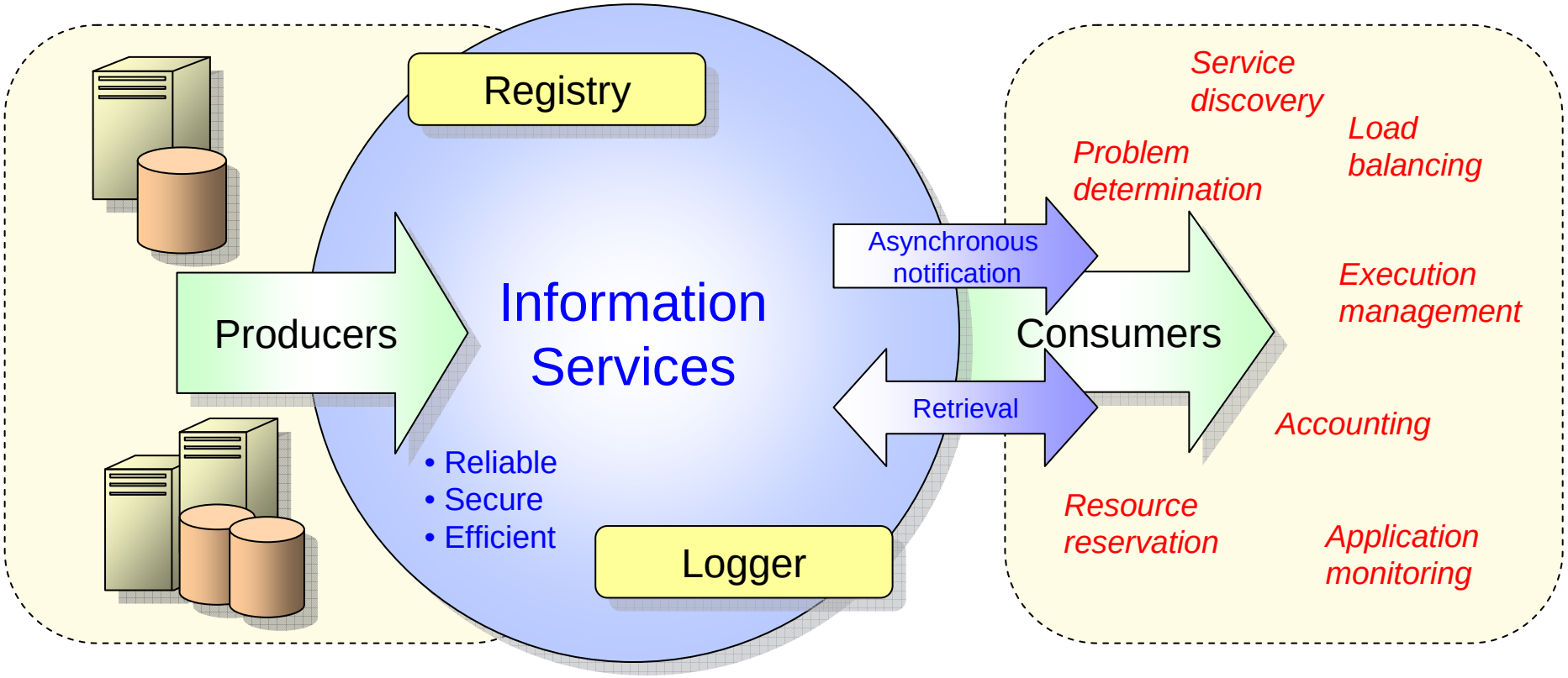
Resource Management

- Provides a framework to integrate resource management functions
  - interfaces, services, information models, etc.
- Enables integrated discovery, monitoring, control, etc.



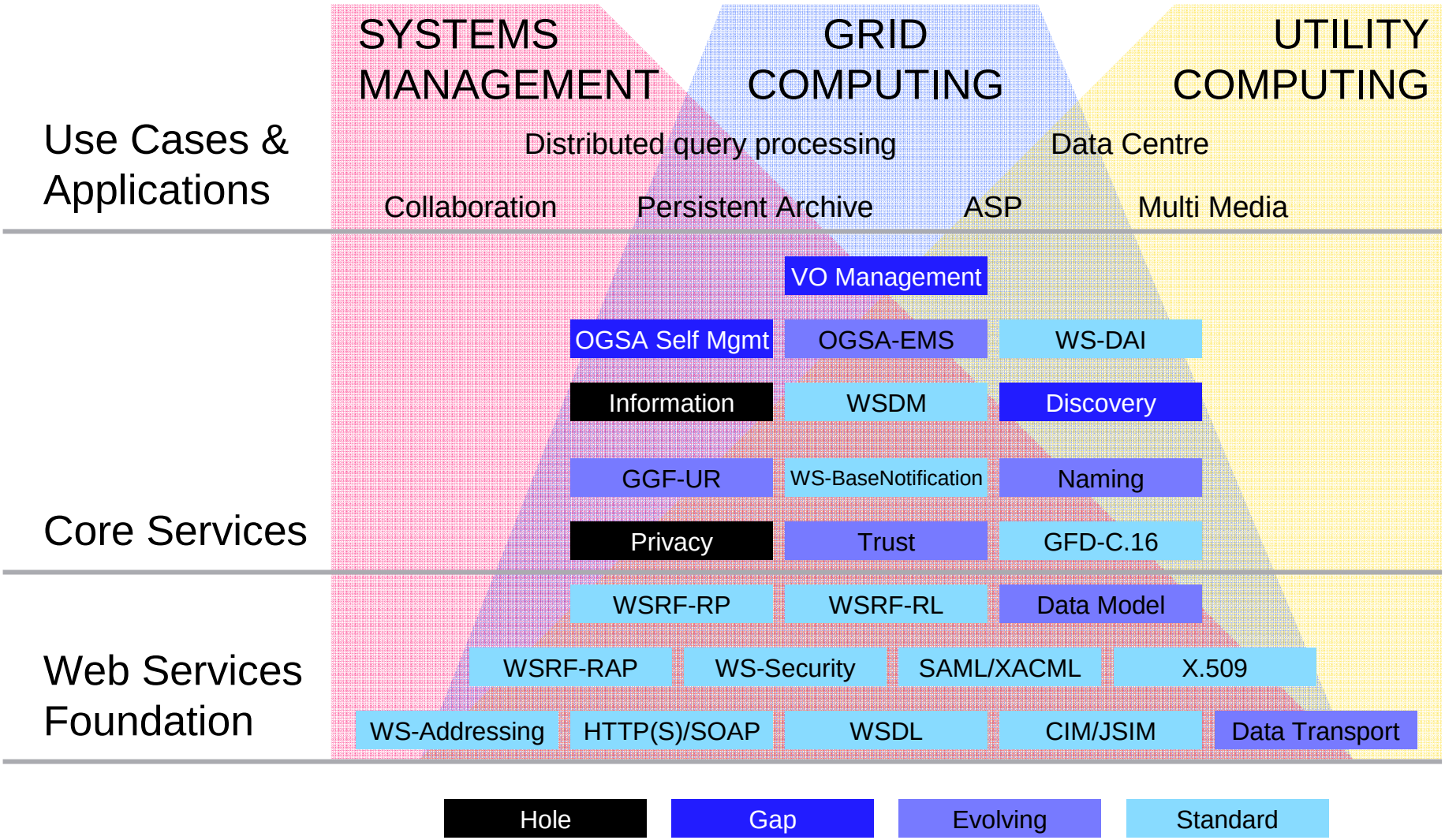
Information Services

Provide **management** and **access** facilities for **information** about applications and resources in the grid environment



Specifications Landscape: April 2006

Warning: Volatile data!



Summary & Conclusions

Grids

- Many reasons motivating investment in grids
  - Collaboration for Global Science & Business
  - Resource integration & sharing
- New approach to large scale distributed systems
  - Large coordinated effort
  - Industry & Academia
- Many technical and socio-economic challenges
  - Work for you all
- Many new opportunities
  - Work for you all

Summary: Take home message

- E-Infrastructure is arriving
  - Built on Grids & Web Services
  - Data and Information grow in importance
- There is a dramatic rate of change
- An opportunity for everyone

Can you ride the wave?

Questions & Comments Please