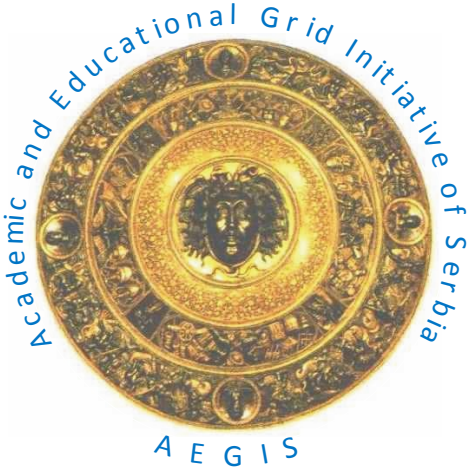
# Grid and High Performance Computing in Physics

**Antun Balaz**
**Scientific Computing Laboratory**
**Institute of Physics Belgrade, Serbia**
**http://www.scl.rs/**








15 Jul 2012

# Overview

- Why serial is not enough
- Computing architectures
- Parallel paradigms
- Message Passing
- Grid computing

# Serial computing

- Using a single computer to complete a single task
  - concurrent computing
- To improve performance
  - Optimize program code
  - Use mathematical libraries
  - Improve the hardware
    - Moore's law - empirical observation made in 1965 that the number of transistors on an integrated circuit for minimum component cost doubles every 24 months.
    - Bigger, faster and more memory (DDR3, FBDIMMS)
    - More storage!

# Why serial is not enough

- Realistic simulations require really really
  - Large numbers of particles
  - Large MC samples
  - Large statistics
  - Combinatorially large spaces to be searched
  - Excessively fine multidimensional discretizations
  - Huge data inputs to be processed
  - …
- We want to solve problems harder, faster, better, stronger!
- Parallel hardware is available (clusters)
- Parallel software is available (libraries)
- And we want to learn something new…

# Modern computing architectures

- Shared memory (SMP)
  - Single large system where all CPUs can access the whole available memory
- Distributed memory
  - Each CPU can access only local memory attached to it (nodes with one single-core CPU)
- Hybrid systems (majority of clusters)
  - Nodes with several single-core CPUs
  - Nodes with a single multicore CPU
  - Nodes with several multicore CPUs

# Parallel paradigms (1)

- The two (three) architectures determine two basic paradigms
  - Data parallel (shared memory)
    - Single memory view, all processes (usually threads) could **directly access the whole memory**
  - Message Passing (distributed memory
    - All processes could **directly access only their local memory**

# Parallel paradigms (2)

| Programming environments | |
|---|---|
| **Message Passing** | **Data Parallel** |
| Standard compilers | Ad hoc compilers |
| Communication libraries | Source code directive |
| Ad hoc commands to run program | Standard Unix shell to run program |
| Standard: MPI | Standard: OpenMP |

# Parallel paradigms (3)

| Architecture | |
|---|---|
| Distributed memory | Shared memory |
| **Programming paradigm** | |
| Message passing | Data parallel |
| **Programing model** | |
| Domain decomposition | Functional decomposition |

# Programming models

- Domain decomposition
  - Data divided into equal chunks and distributed to available CPUs
  - Each CPU process its own local data
  - Exchange of data if needed
- Functional decomposition
  - Problem decomposed into many sub-tasks
  - Each CPU performs one of sub-tasks
  - Similar to server/client paradigm

# Flynn's taxonomy

- SPSD (Single program, single data)
- **SPMD (Single program, multiple data)**
  - not synchronized at individual operation level
  - equivalent to MIMD since each MIMD program can be made SPMD
- **MPSD (Multiple program, single data)**
- **MPMD (Multiple program, multiple data)**

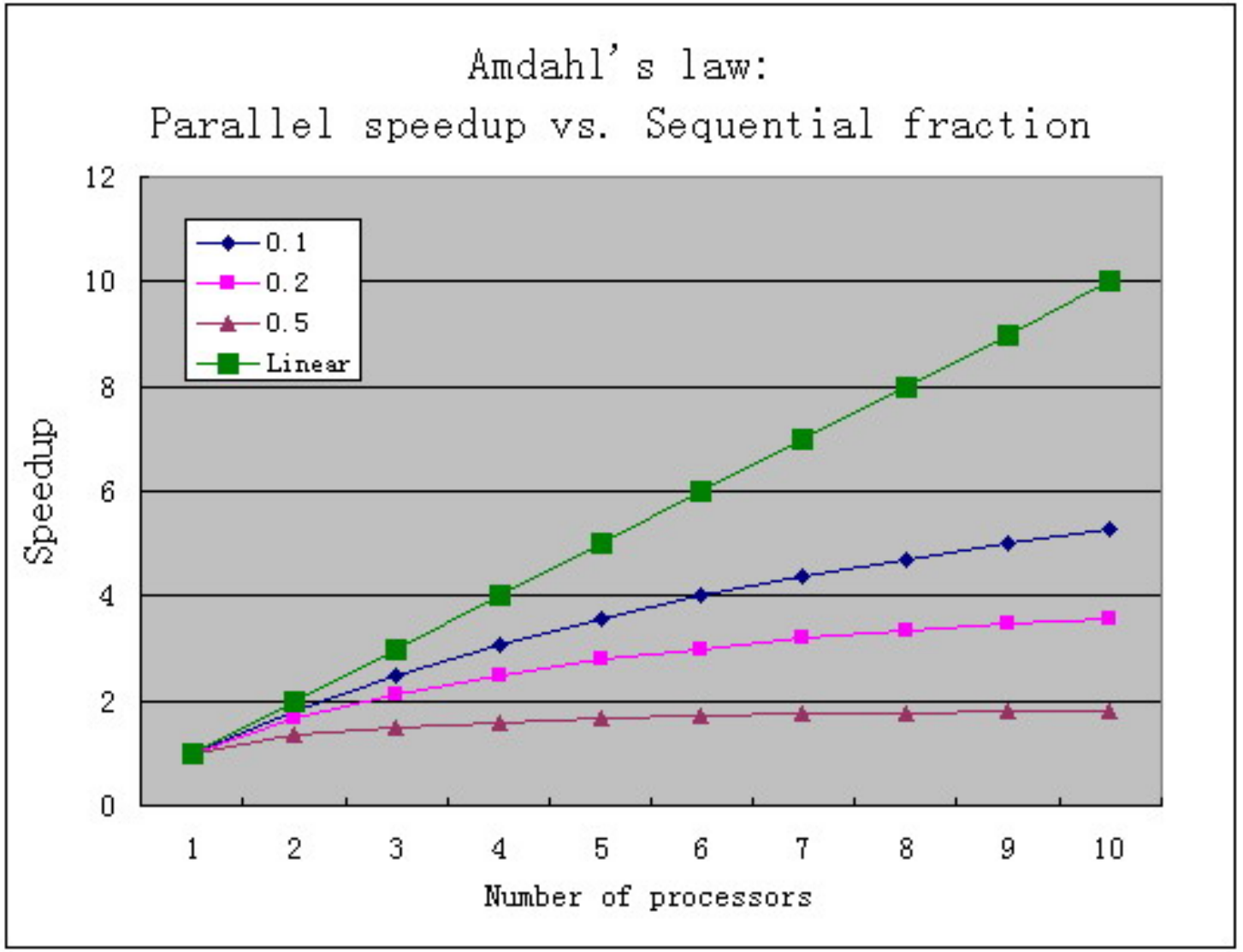
# Parallel paradigms (4)

| Model | Paradigm | Flynn's taxonomy |
|---|---|---|
| Domain decomposition | Message Passing | **SPMD** |
| | Data Parallel - HPF | |
| Functional decomposition | Data Parallel - OpenMP | **MPSD** |
| | Message Passing | **MPMD** |

# Parallelism requires…

- Balancing of the load
  - Applies to computation, I/O operations, network communication
  - Relatively easy for domain decomposition, not so easy for functional decomposition
- Minimizing communication
  - Join individual communications
  - Eliminate synchronization – the slowest process dominates
- Overlap of computation and communication
  - This is essential for true parallelism!

# Effective parallel performance



# Message Passing

- Parallel programs consist of separate processes, each with its own address space
  - Programmer manages memory by placing data in a particular process
- Data sent explicitly between processes
  - Programmer manages memory movement
- Collective operations
- On arbitrary set of processes
- Data distribution
  - Also managed by the programmer

# What is MPI?

- Message Passing Interface
- A message-passing library specification
  - extended message-passing model
  - not a language or compiler specification
  - not a specific implementation or product
- For parallel computers, clusters, and heterogeneous networks
- Full-featured
- Designed to provide access to advanced parallel hardware for end users, library writers, and tool developers

# MPI references

- The Standard itself:
  - at http://www.mpi-forum.org
  - All MPI official releases, in both postscript and HTML
- Other information on Web:
  - at http://www.mcs.anl.gov/mpi
  - pointers to lots of stuff, including talks and tutorials, a FAQ, other MPI pages

# MPI Implementations

- Because MPI is a standard, there are several implementations
- MPICH - http://www-unix.mcs.anl.gov/mpi/
  - Freely available, portable implementation
  - Available on everything
- OpenMPI - http://www.open-mpi.org/
  - Includes the once popular LAM-MPI
- Vendor specific implementations
  - CRAY, SGI, IBM

# When do you need MPI?

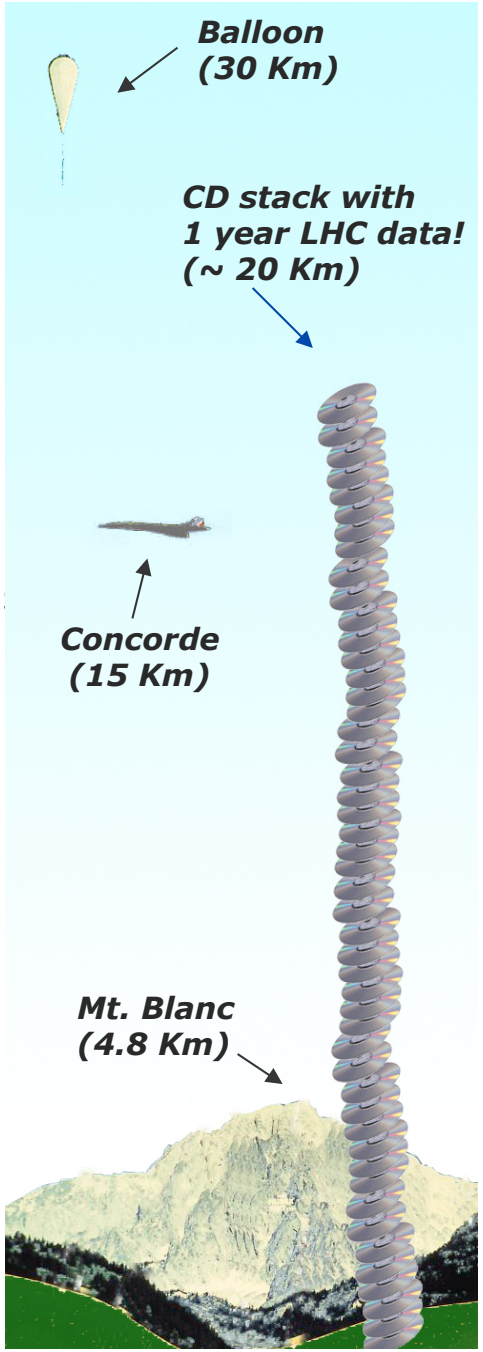
- You need a portable parallel program
- You are writing a parallel library
- You have irregular or dynamic data relationships that do not fit a data parallel model
- You care about performance

# Writing an MPI program

- MPI is a library
- All operations are performed with function (subroutine) calls
- Basic definitions are in
  - mpi.h for C/C++
  - mpif.h for Fortran 77 and 90
  - MPI module for Fortran 90 (optional)

# Grid - Motivation

- **Why the Grid?**
- Science is becoming increasingly digital and needs to deal with increasing amounts of data
- Particle Physics and other disciplines
  - Large amount of data produced
  - Large worldwide organized collaboration
  - e.g. Large Hadron Collider (LHC) at CERN
    - 40 million collisions per second
    - ~10 petabytes/year (~10 Million GBytes)

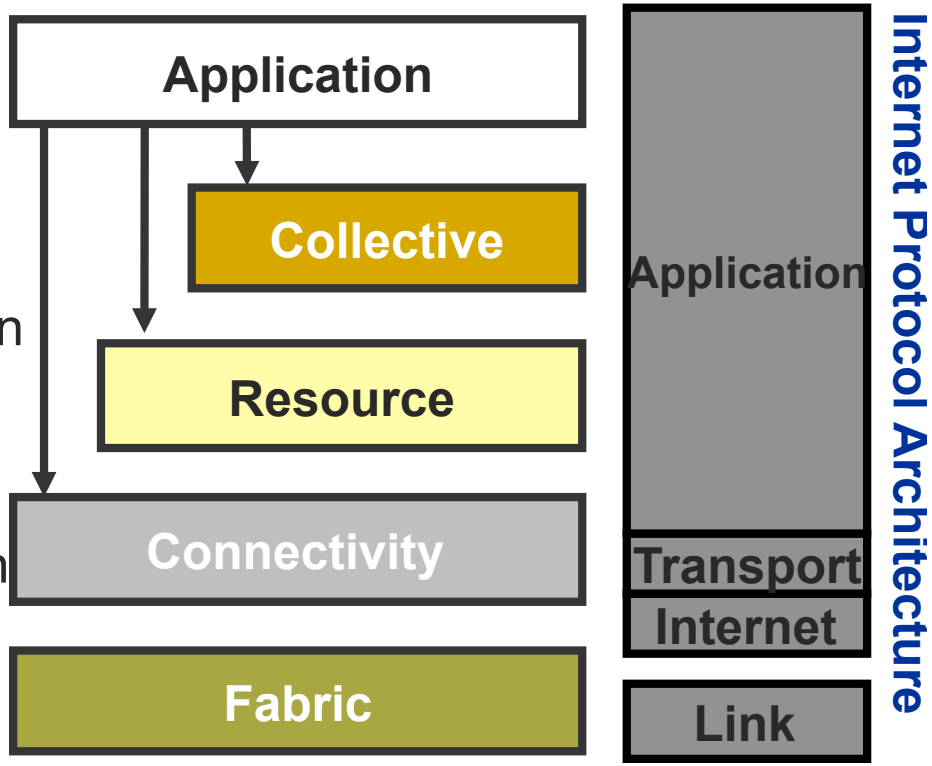


# Solution: The Grid

… securely share distributed resources (computation, storage, etc) so that users can collaborate within Virtual Organisations (VO)

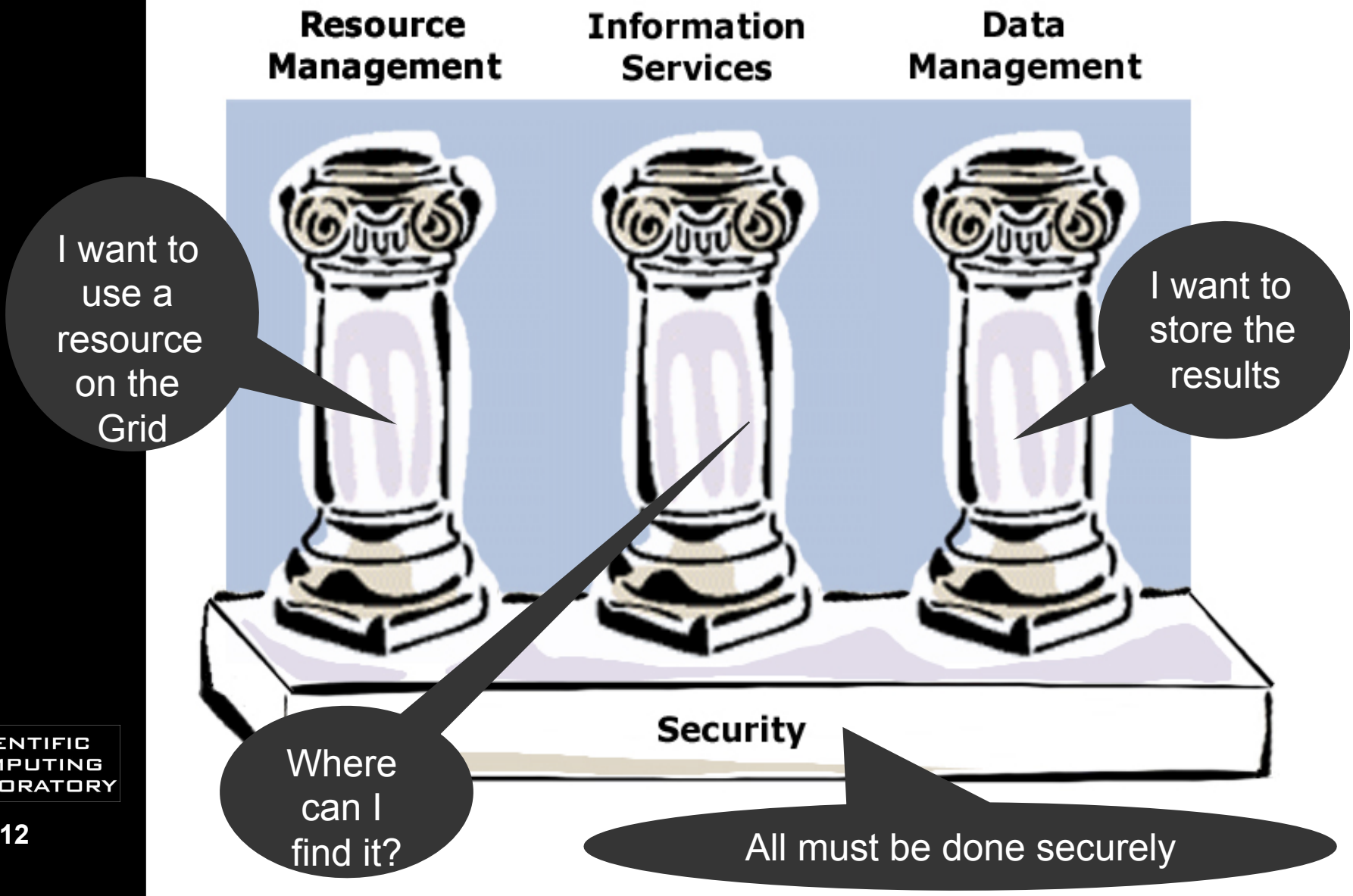
# The Grid stack

- Application layer
  - Grid programs
- Collective layer
  - Resource Co-allocation
  - Data Management
- Resource layer
  - Resource Managemen
  - Information Services
  - Data Access
- Connectivity layer
  - Grid Security Infrastructure
  - High-performance data transfer protocols
- Fabric layer
  - the hardware: computers (parallel, clusters..), data storage servers



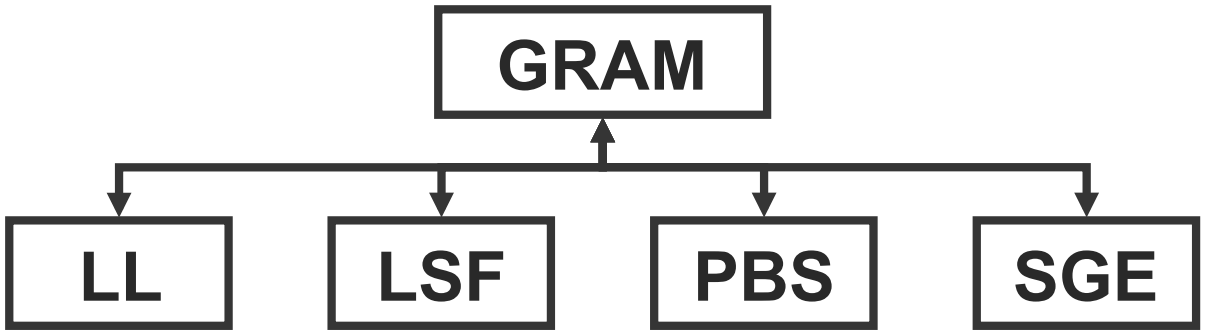
# Grid foundations

- Defined by the Globus: http://globus.org



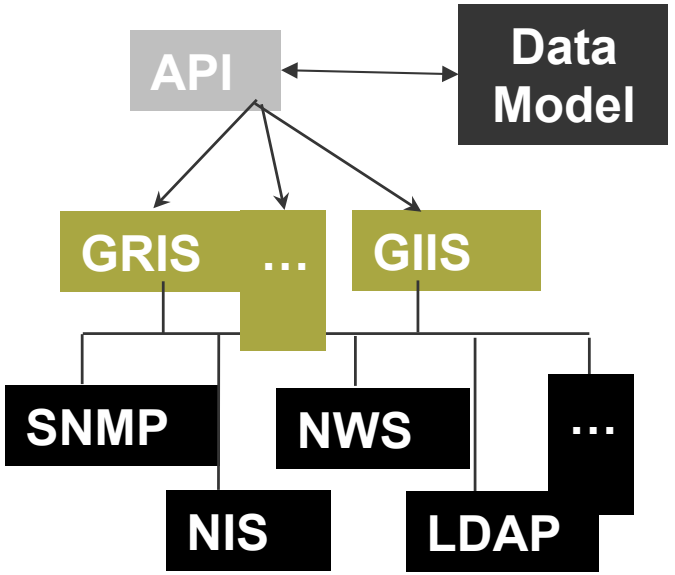
# Resource Management

- Everything (or anything) is a resource
  - Physical or logical (single computer, cluster, parallel, data storage, an application...)
  - Defined in terms of **interfaces**, not devices
- Each site must be autonomous (local system administration policy)
- Grid Resource Allocation Manager (GRAM)
  - Defines resource layer protocols and APIs that enable clients to **securely instantiate a Grid computational task** (i.e. a job)
  - Secure remote job submissions
  - Relies on local resource management interfaces



# Information Services

- Maintains information about hardware, software, services and people participating in a Virtual Organization
  - Should scale with the Grid´s growth
- "Find a computer with at least 2 free CPUs and with 10GB of free disk space..."

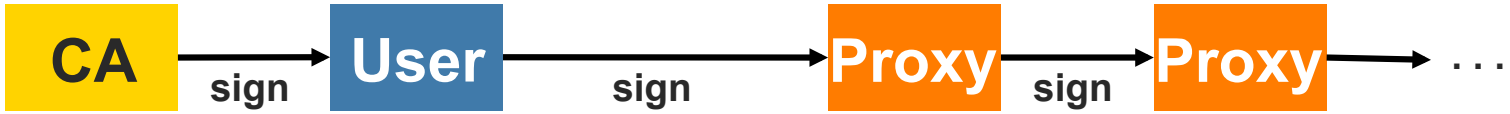


# Data Management

- Data access and transfer
  - Simple, automatic multi-protocol file transfer tools: **Integrated with Resource Management service**
    - Move data from/to local machine to remote machine, where the job is executed (staging – stageout)
    - Redirect stdin to a remote location
    - Redirect stdout and stderr to the local computer
    - Pull executable from a remote location
  - To have a secure, high-performance, reliable file transfer over modern WANs: **GridFTP**

# Security

- Basic security:
  - **Authentication:** Who we are on the Grid?
  - **Authorization:** Do we have access to a resource/ service?
  - **Protection:** Data integrity and confidentiality
- but, there are thousands of resources over different administration domains...:
  - **Single sign-on**, i.e. give a password once, and be able to access all resources (to which we have access)
- Grid Security Infrastructure (GSI):
  - **Grid credentials:** digital certificate and private key
    - Based on PKI X.509 standard
    - CA signs certificates. Trust relationship
  - **Proxy certificates:** Temporary self-signed certs, allowing single sign-on: Proxy delegation



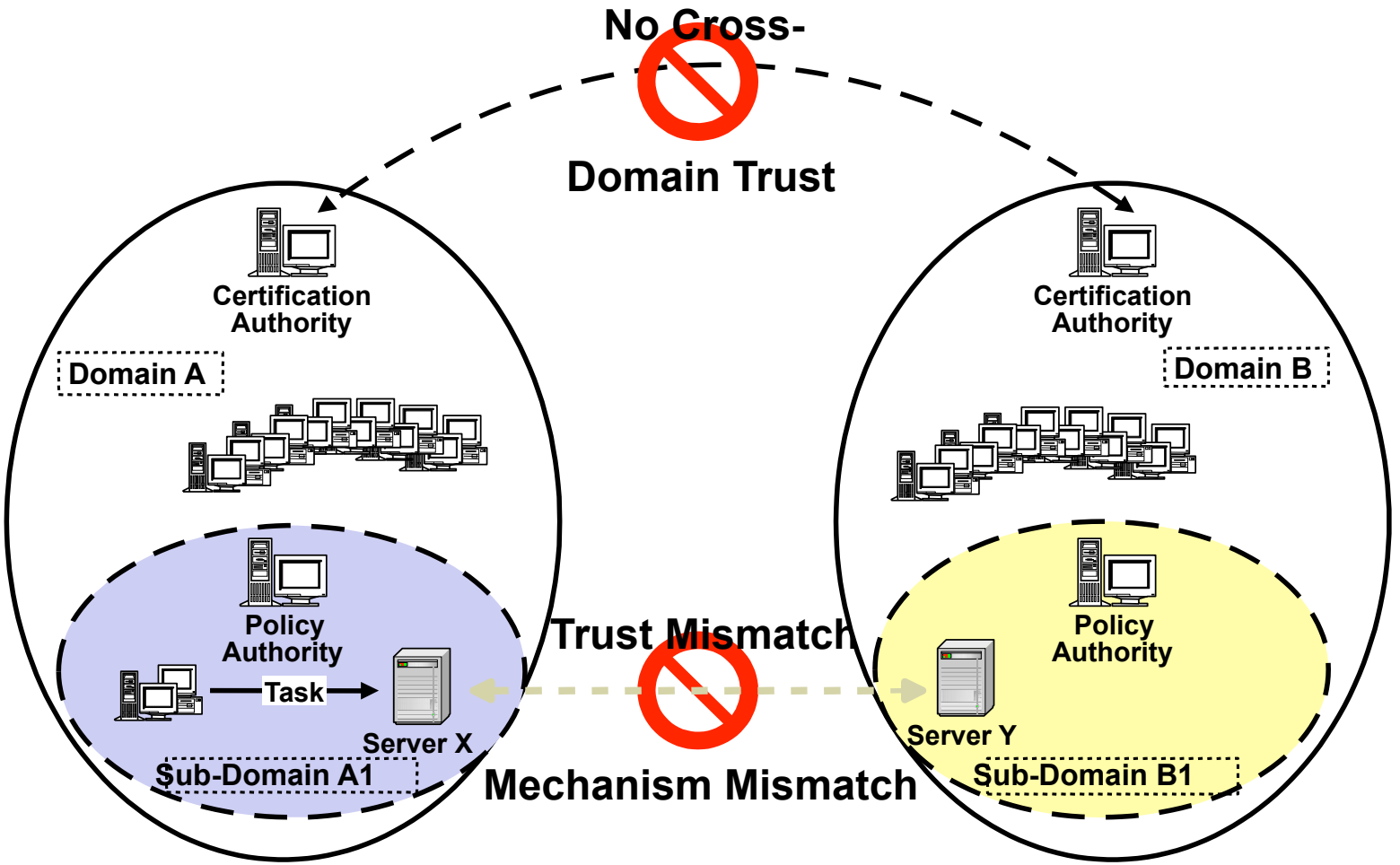
# gLite – **Grid middleware**

- The Grid relies on advanced software – the middleware - which interfaces between resources and the applications
- The GRID middleware
  - Finds convenient places for the application to be executed
  - Optimises use of resources
  - Organises efficient access to data
  - Deals with authentication to the different sites that are used
  - Run the job & monitors progress
  - Transfers the result back to the scientist

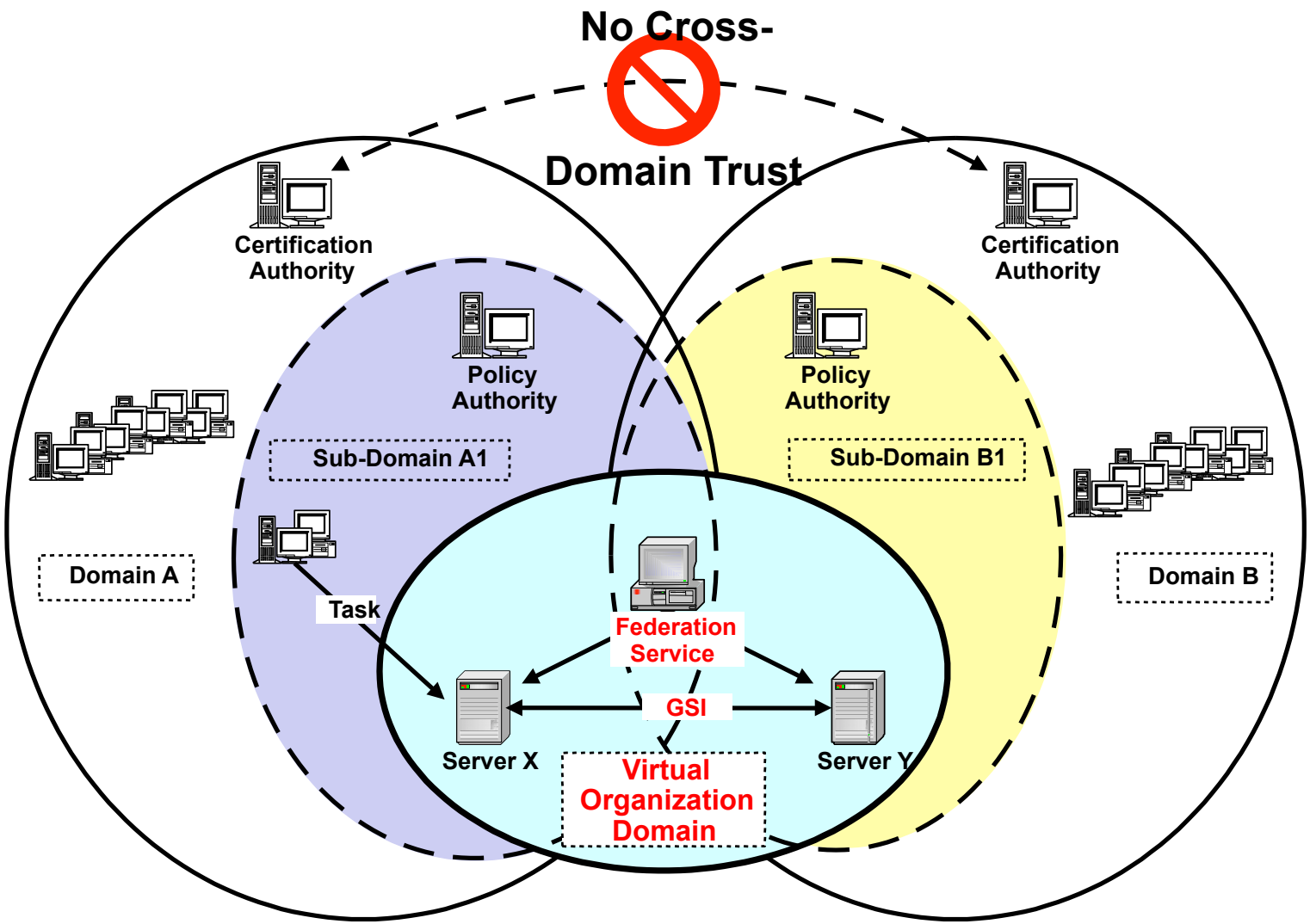
# Set of basic Grid services

- Job submission/management
- File transfer (individual, queued database access)
- Data management (replication, metadata)
- Monitoring/Indexing system information

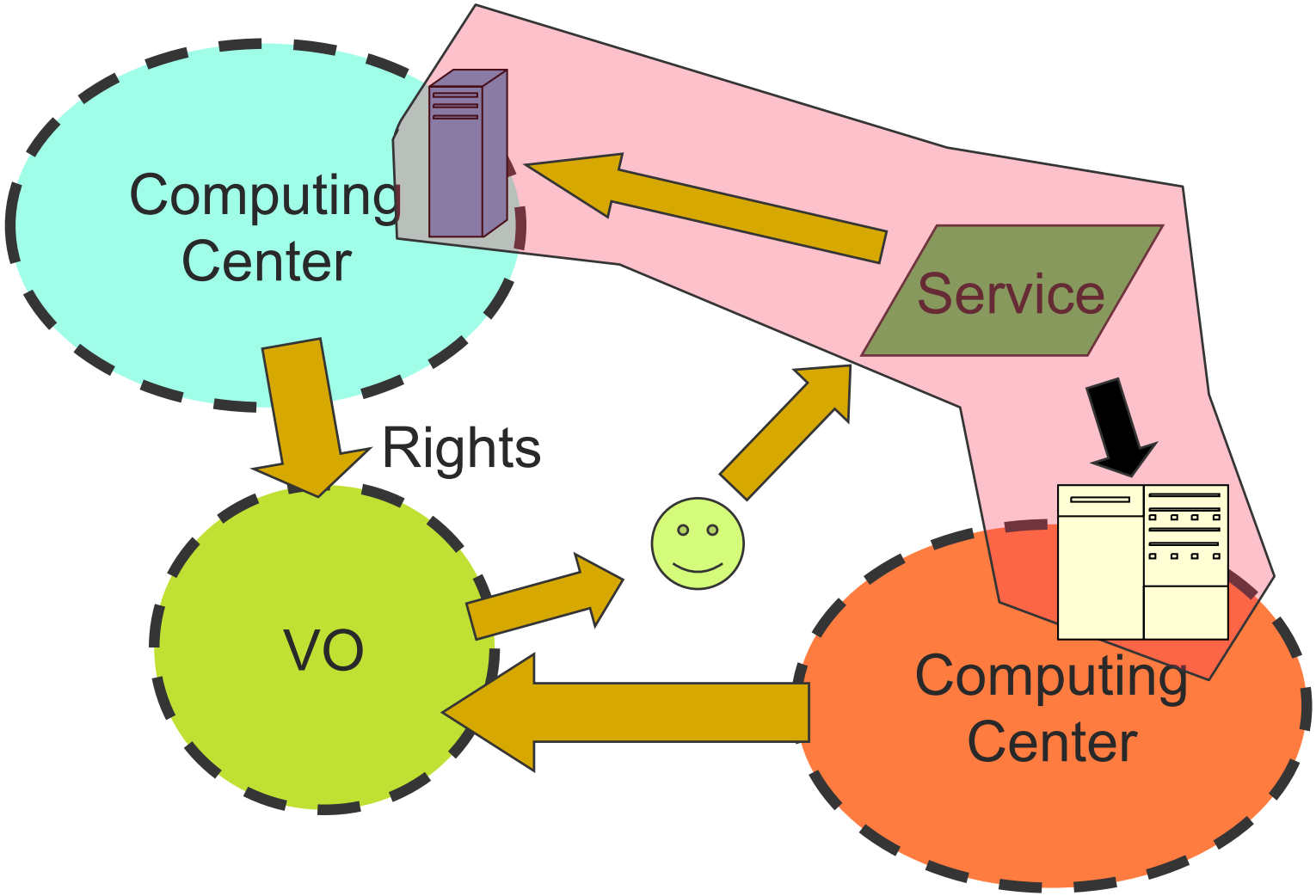
# Multi-institution issues



# Grid solution: use of VOs



# Use delegation to establish dynamic distributed system



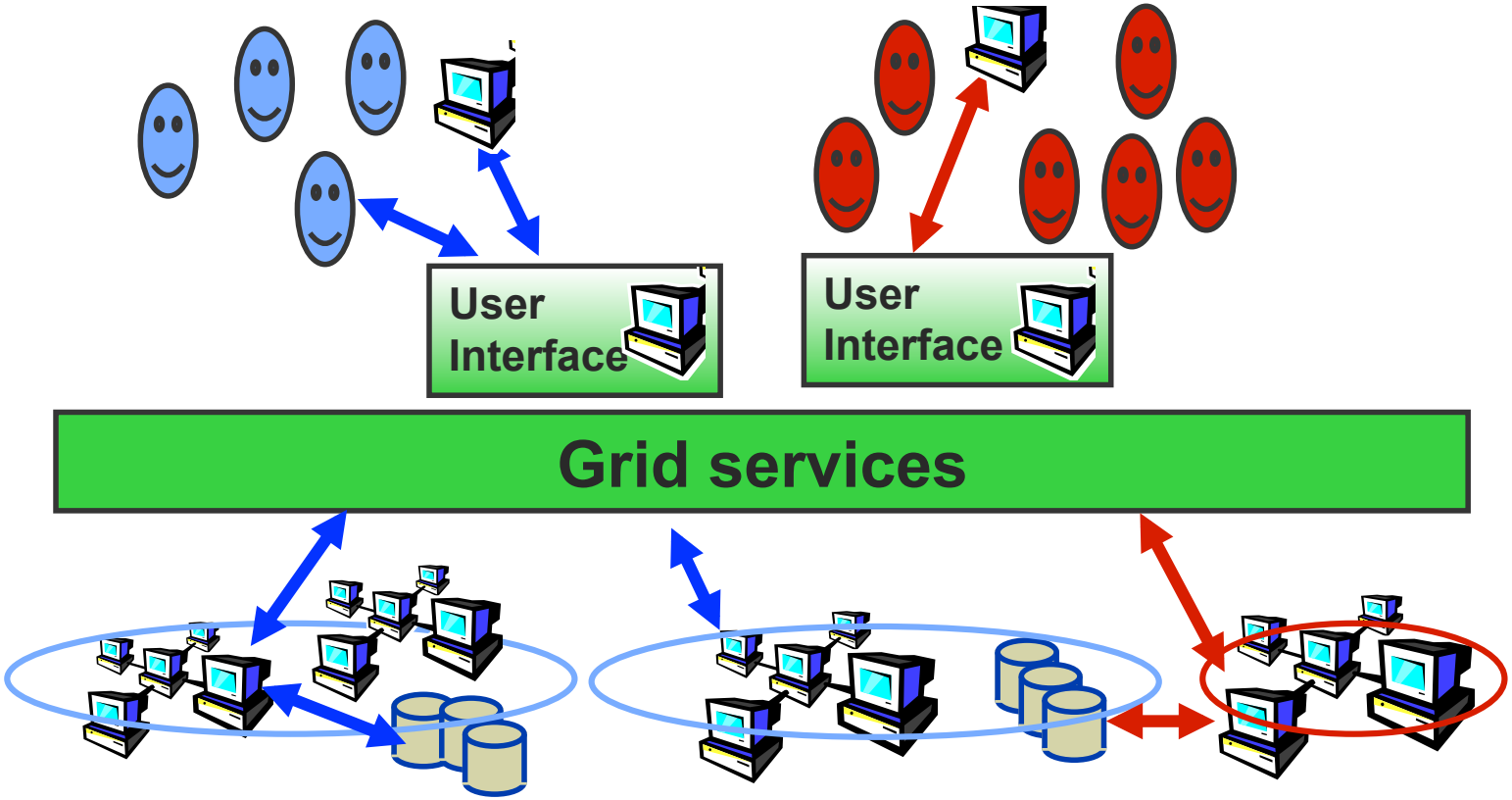
# Grids and VOs (1)

- Virtual organizations (VOs) are groups of Grid users (authenticated through digital certificates)
- VO Management Service (VOMS) serves as a central repository for user authorization information, providing support for sorting users into a general group hierarchy, keeping track of their roles,etc.
- VO Manager, according to VO policies and rules, authorizes authenticated users to become VO members

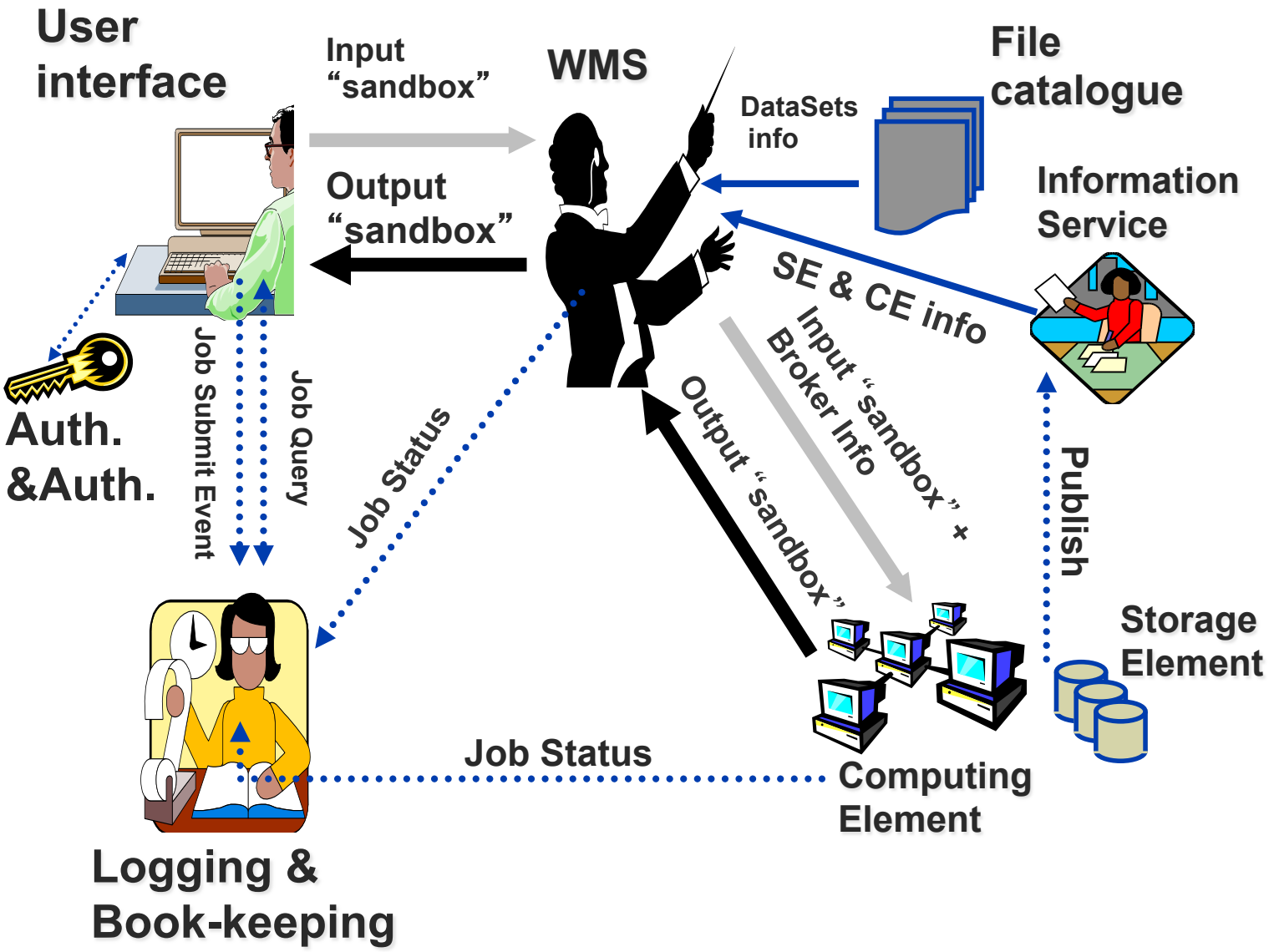
# Grids and VOs (2)

- Resource centers (RCs) may support one or more VOs, and this is how users are authorized to use computing, storage and other Grid resources
- VOMS allows flexible approach to A&A on the Grid

# User view of the Grid



# What really happens




SCIENTIFIC COMPUTING LABORATORY

15 Jul 2012

# Grid in a nutshell

- Grid structure is complicated but hidden from end-users, enabling all the comfort they need
- Users just need to join the VO and obtain certificates: we already have some VOs at hand for you!
- Use of Grid is then just as easy as the use of a typical Linux cluster