# A brief introduction to GRID COMPUTING

**Stefano Cozzini**

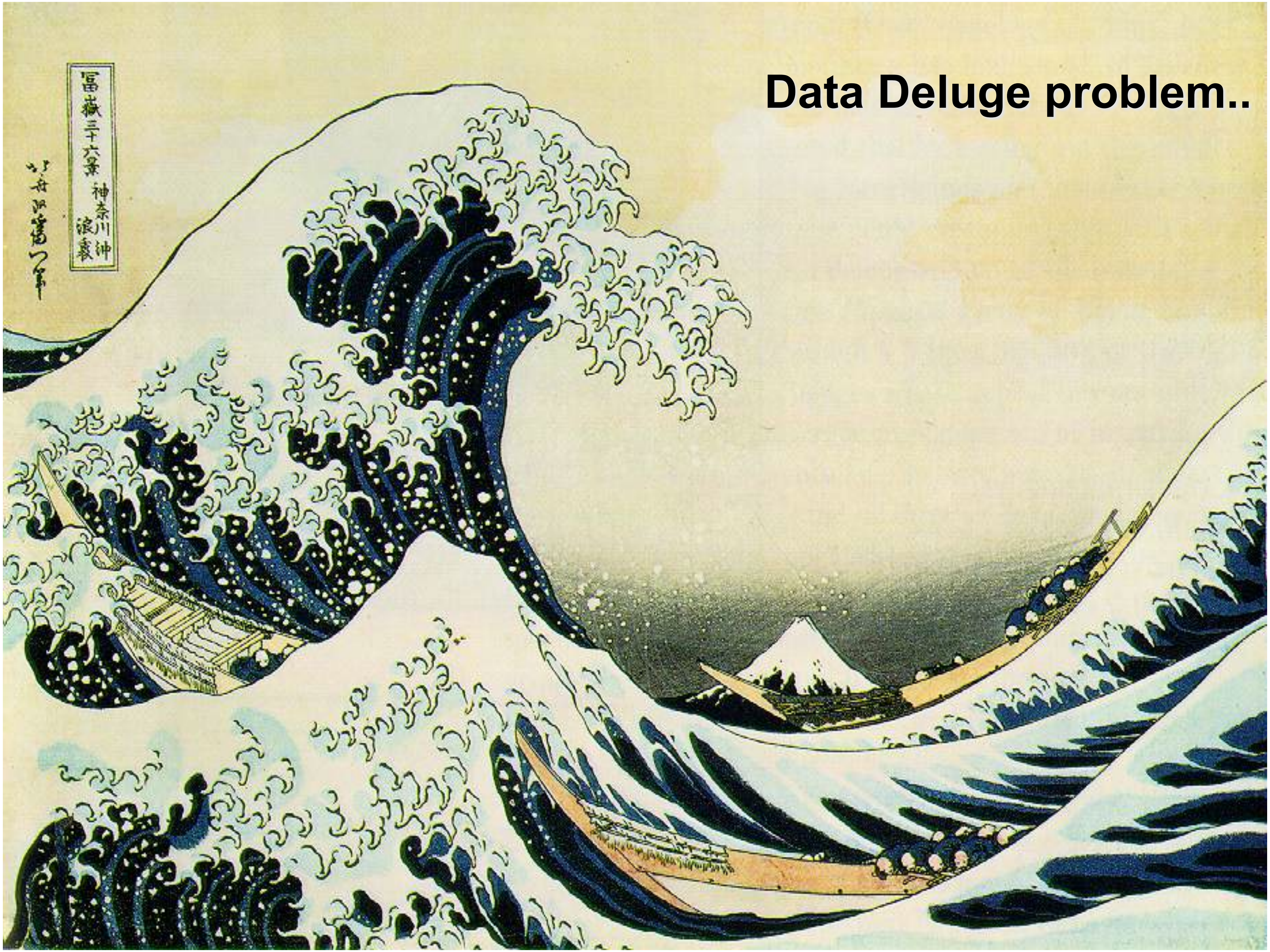
Democrito and SISSA/eLAB - Trieste

## New challenges in Science

- Going further in scientific knowledge
  - New high sensitivity sensors and instruments
  - Globally distributed collaborations

- Delocalized knowledge
  - Scientific and technical knowledge is “distributed”
  - Laboratories are distributed
  - Scientific data are distributed

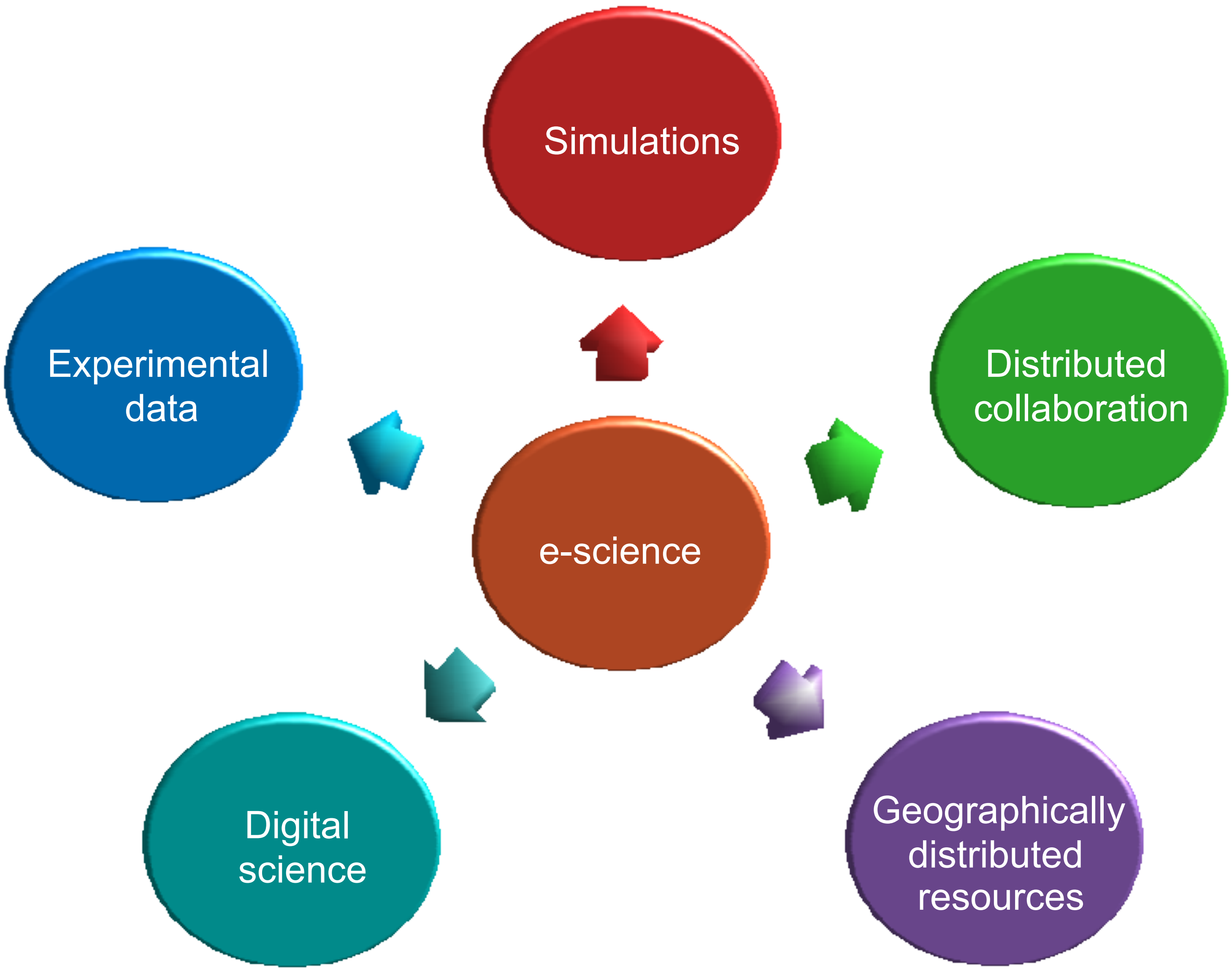
# Data Deluge problem..



# e-science

“eScience is about global collaboration in key areas of science and the next generation of infrastructure that will enable it.”

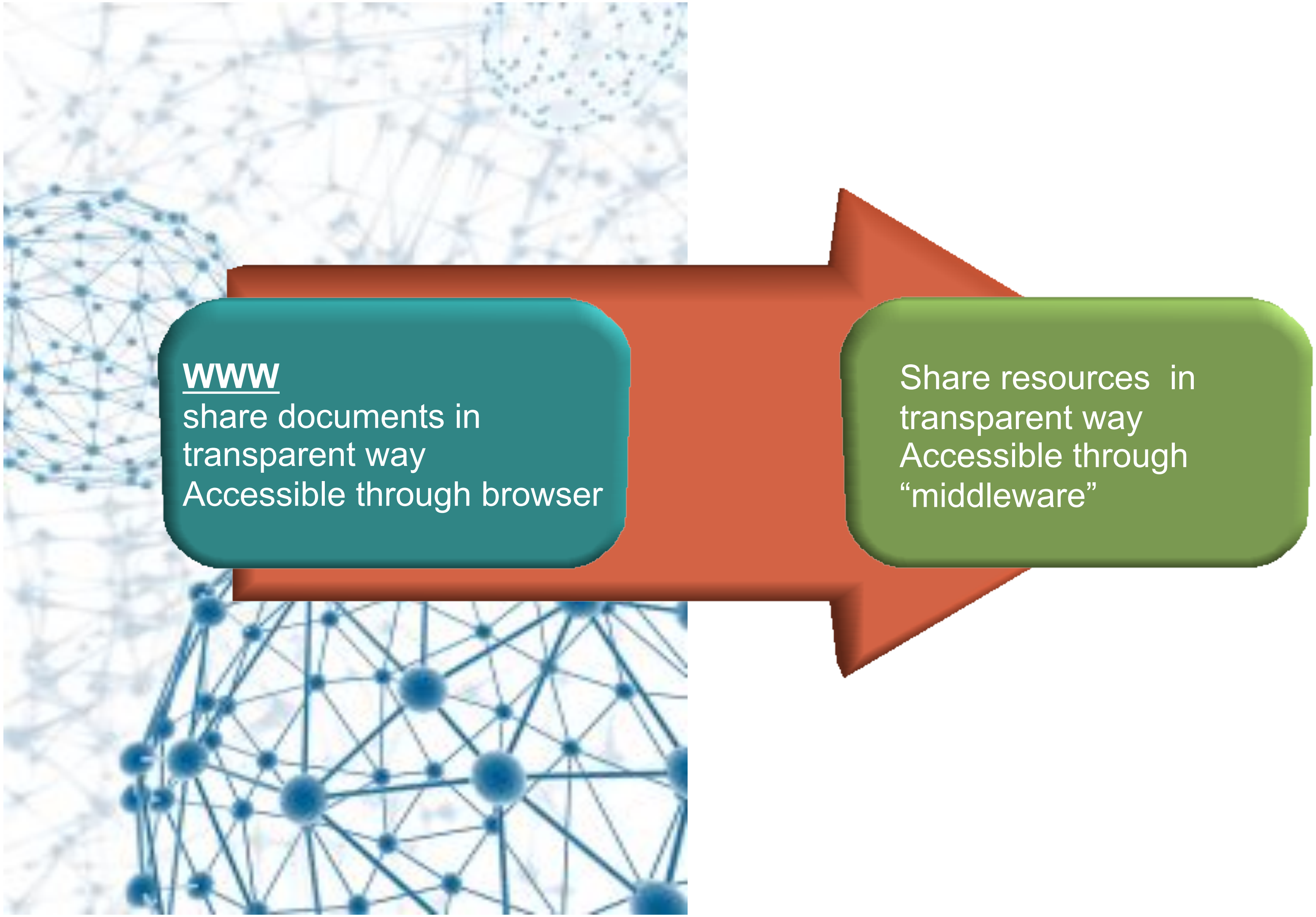
*Dr.John Taylor, Director General of the Research Councils 1998-2003*


Simulations
Experimental data
Distributed collaboration
e-science
Digital science
Geographically distributed resources

## Using internet to make science

- On-line publication paper/pre-prints (eg. babbage.sissa.it)
- CPU cycle scavenging (eg. Seti@home, Condor)
- Sloan Digital  Sky Survey: online database of astronomical data http://www.sdss.org/

# A new paradigm



# What is your paradigm?

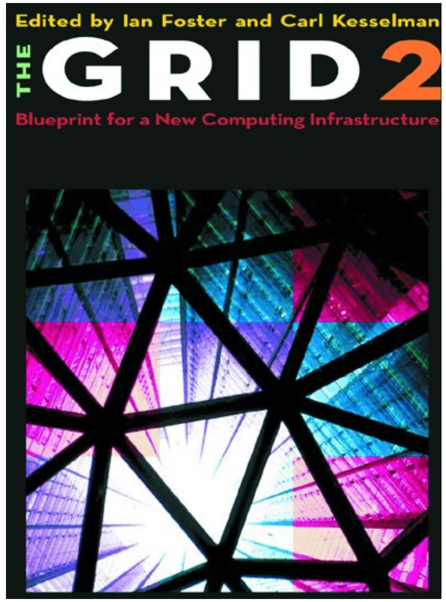
**Parallel Computing**
single systems with many processors working on same problem

**Distributed Computing**
many systems loosely coupled by a scheduler to work on related problems

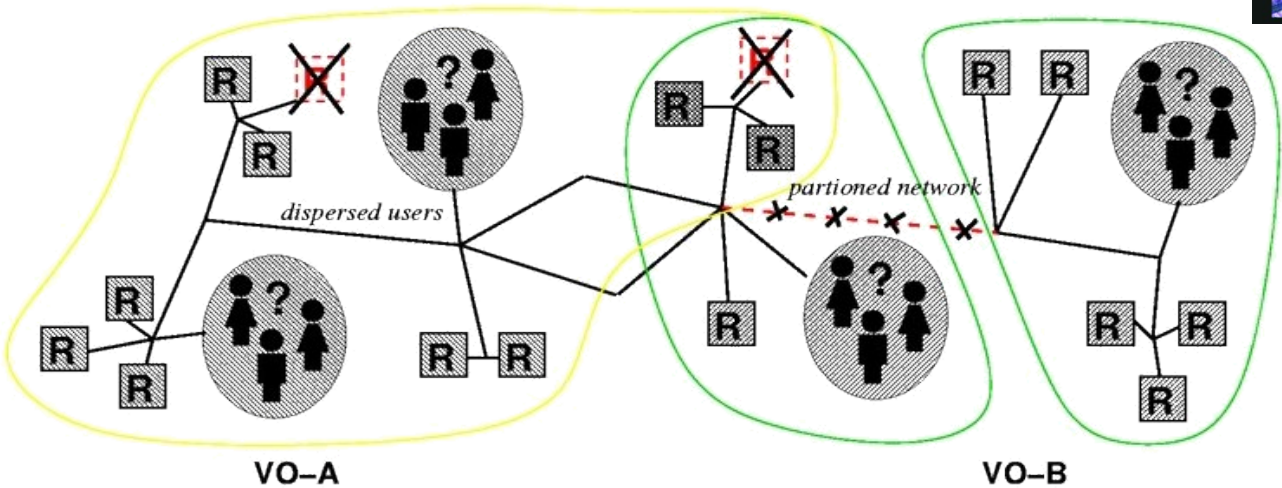
**Grid Computing**
many systems tightly coupled by software, perhaps geographically distributed, to work together on single problems or on related problems

# What is Grid Computing?

# The Grid




"Resource sharing & coordinated problem solving in dynamic … virtual organizations"




1. Enable integration of distributed service & resources
2. Using general-purpose protocols & infrastructure
3. To achieve useful qualities of service

"The Anatomy of the Grid", Foster, Kesselman, Tuecke, 2001

# The Grid Problem

- Flexible, secure, coordinated sharing of computation among dynamic collections of individuals, institutions, and resources
- Enable communities ("virtual organizations") to share geographically distributed resources as they pursue common goals -- assuming the absence of…
  - central location
  - central control
  - omniscience
  - existing trust relationships

**The Anatomy of the Grid: Enabling Scalable Virtual Organizations.** I. Foster, C. Kesselman, S. Tuecke. *International J. Supercomputer Applications*, 15(3), 2001.


Resource sharing, common goals
Fault Tolerant
Linked by networks, cross admin domains
VO
Dynamic
Geographically distributed resources people

## The Programming Problem

- Applications require resources (compute power, storage, data, instruments, displays) at many sites for many users.
- Some requirements:
  - Abstractions and models to increase speed/robustness/ etc. of development
  - Tools to ease application development and diagnose common problems, ease deployment
  - Code/tool sharing to allow reuse of code components developed by others

# Grid must support computational workflows

- Locate “suitable” computers
- Authenticate with appropriate sites
- Allocate resources on those computers
- Initiate computation on those computers
- Configure those computations
- Select “appropriate” communication methods
- Compute with “suitable” algorithms
- Access data files, return output
- Respond “appropriately” to resource changes

# Grid Requirements

- identity & authentication
- authorization & policy
- resource/service discovery
- resource allocation
- (co-)reservation, workflow
- remote data access
- rapid data transfer
- monitoring
- intrusion detection
- resource management
- accounting
- fault management
- system evolution
- and more…

# Grid Computing - Functions

- Grid computing must provide typically these basic functions (Foster/Kesselman)
  - resource discovery and information collection & publishing
  - data management on and between resources
  - process management on and between resources
  - common security mechanism underlying the above
- In addition, it should include:
  - process and session recording/accounting

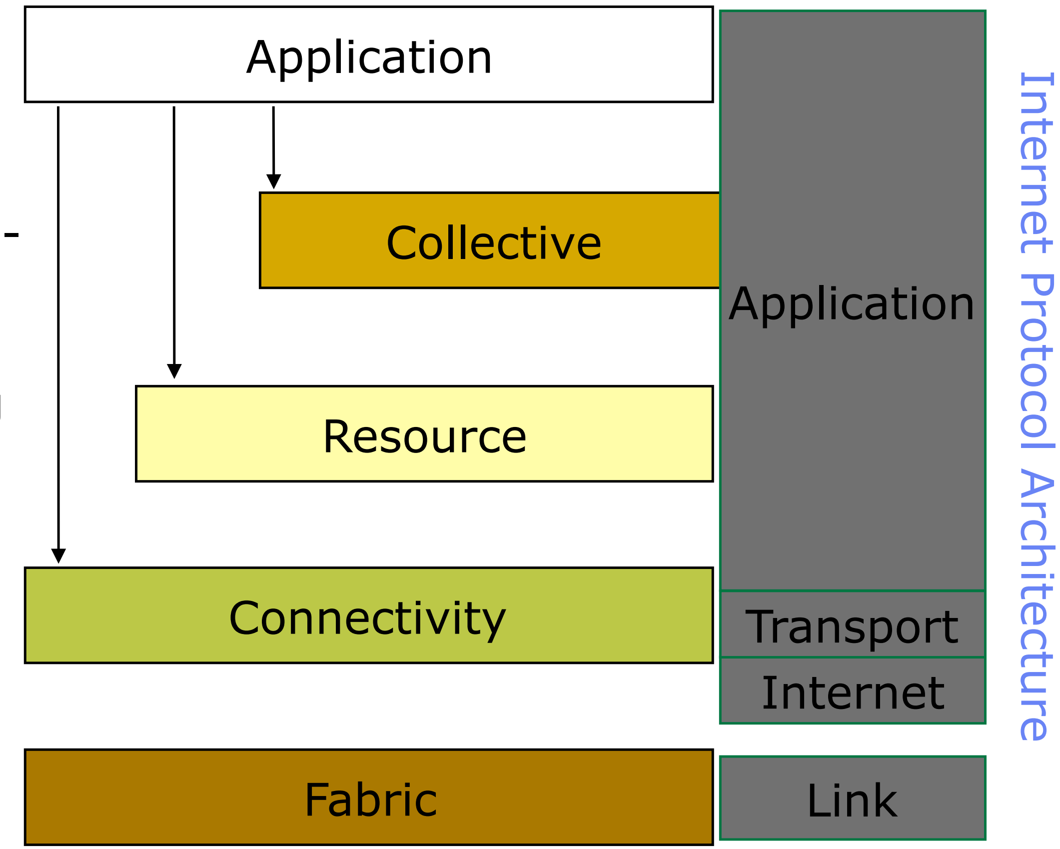
# Layered Grid Architecture
# (By Analogy to Internet Architecture)

“Coordinating multiple resources”: ubiquitous infrastructure services, app-specific distributed services

“Sharing single resources”: negotiating access, controlling use

“Talking to things”: communication (Internet protocols) & security

“Controlling things locally”: Access to, & control of, resources




Slide courtesy of C. Kessleman Cal(IT)2 Presentation

# Layered Grid Architecture

- Fabric Layer - provides the local services of a resource:
    - computational, storage, network
- Connective Layer - core communication and authentication protocols
    - Enables exchange of data between fabric layer resources
    - Security and authentication important here

# Layered Grid Architecture (cont.)

- Resource Layer – enables resource sharing
  - Builds on connectivity layer to control and access resources (Ex: data servers)
- Collective Layer - coordinates interactions across multiple resources
  - Ties multiple resources and services together
  - (Ex: metacatalogues)
- Application Layer - user applications use collective, resource, and connective layers to perform grid operations in a virtual organization

## Some Solutions

- Middleware Toolkits:
  - Condor
  - Globus Toolkit
  - Legion/Avaki
  - Glite
  - Garuda..
  - Condor (now Sun Grid Engine)
  - Unicore
  - Arc
- Higher Level Toolkits
  - JavaCoG
  - GridPortal Toolkit, Grid Portal Development Toolkit (GPDK)
  - Vine
  - Condor-G
  - SGE

NEED OF COMMON STANDARD AND INTEROPERABILITY SOLUTIONS

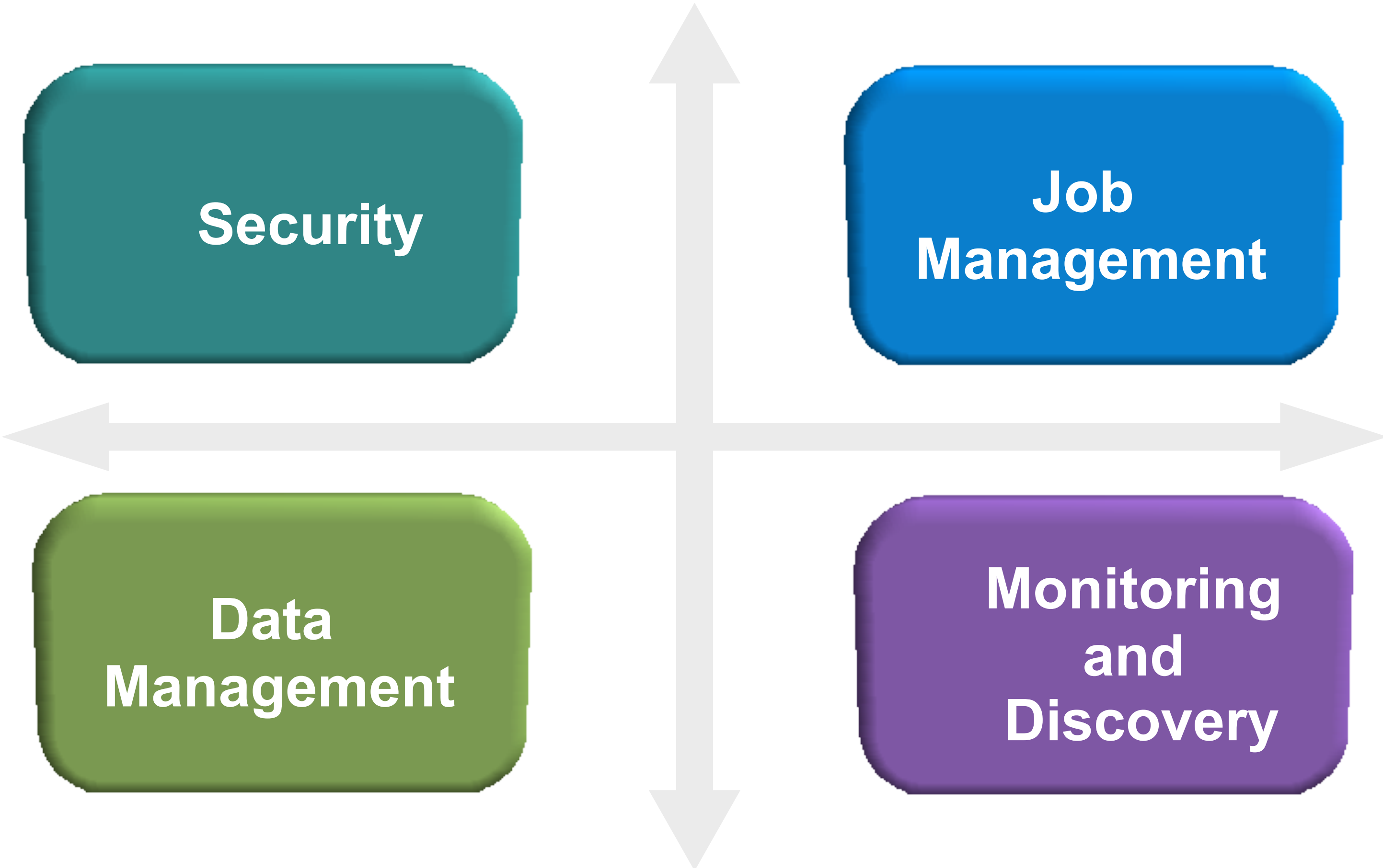
## Middleware: gLite

- gLite is the middleware for grid computing born from the collaborative efforts from academic and industrial research centers as part of the EGEE Project.
- The gLite Grid services follow a *Service Oriented Architecture*
  - **facilitate interoperability among Grid services**
  - **allow easier compliance with upcoming standards**
- Architecture is not bound to specific implementations
  - **services are expected to work together**
  - **services can be deployed and used independently**
- The gLite service decomposition has been largely influenced by the work performed in the LCG project

## Grid Resource

- Storage systems
- Computer clusters
- HPC clusters
- Supercomputers (IBM SP, blue gene, etc)
- Databases
- Keyword: heterogeneous as regards hardware and software

# MW generic services



## Explore gLite middleware

- Bottom-up
  - From low level services to global services
  - From fabric to GRID
  - From Unix user to GRID user

## The Resources

- Group of “sites” glued by the Middleware
- Sites are homogeneous as regards OS and SW:
  - Scientific Linux cern 4
- Sites are heterogeneous as regards HW:
  - x86/x86_64 arch
- Some collective services: WMS, DMS etc.

## A Grid Site

- Computing Element
- Storage Element
- Worker nodes

- Master node
- Storage system
- Computing nodes

- Scheduler+queue system (torque+maui, LSF, etc.

# The Low level services

# Security

- **<u>Grid is a highly complex system</u>**
- Authentication: establishing identity
- Authorization: establishing rights
- Message protection

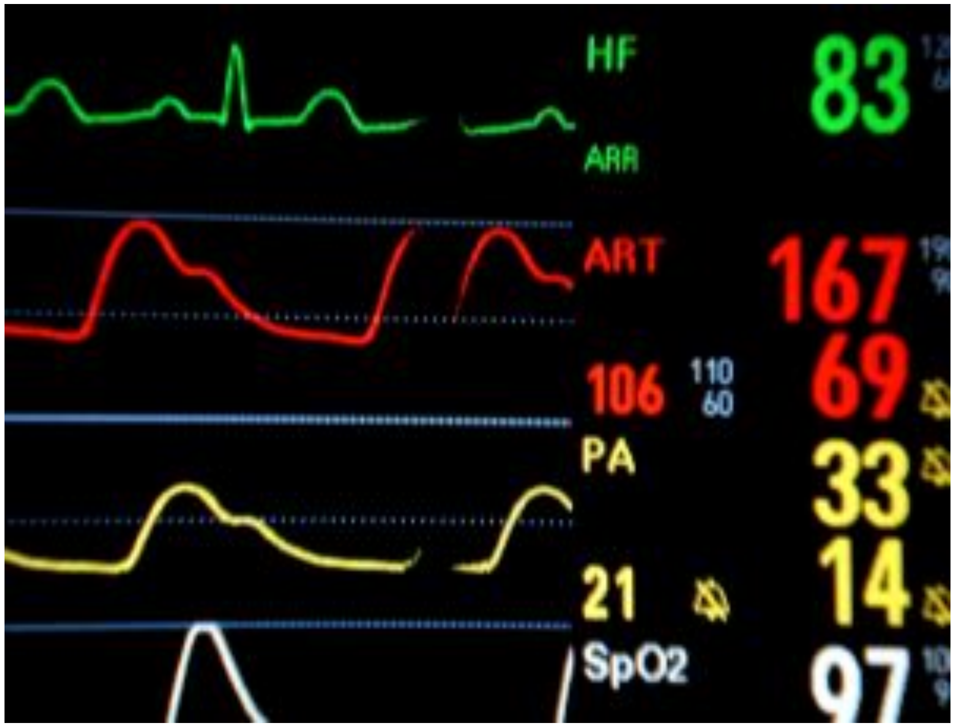
Passwords are not scalable and secure!!!

## What do we require to security?

- Users point of view
  - Easy to use, transparent, single-sign on, no password sharing
- Administrators point of view
  - Define local access control
  - Define local polices
- The Grid Security Infrastructure
  - X509 digital certificates

## Monitor and discovery service

- What is the status of a resource?
- What are the available resources?



## Data Management

- Requirements
  - Fast: as fast as networks and protocols allow
  - Secure: server must only share files with strongly authenticated clients and no passwords in the clear or similar
  - Robust: Fault tolerant, time-tested protocol

⇒ GRIDFTP

# High Level Services

## Information system

- Which resources are available?
- Where are them?
- What is their status?
- How can I optimize their use?

We need a general information infrastructure:
Information System

# Data Management

- Where are data/files?
- Which data/file exist?
- How can I reach it?
- Are they accessible by others?
- ex. LFC file catalogue

# Applications for Grid computing

- Computation intensive
  - Large-scale simulation and analysis ( e.g. atomistic simulations)
  - Engineering (parameter studies, optimization model)
- Data intensive
  - Experimental data analysis (e.g., H.E.P.)
  - Image & sensor analysis (climate)
- Distributed collaboration
  - Online instrumentation (microscopes, x-ray)
  - Remote visualization (climate studies, biology)

# Building your own computational infrastructure

- Open source software + commodity off the shelf hardware provides now tools to build low cost HPC infrastructure
  - based on clusters
- GRID infrastructures are just two clicks away
  - they can provide a looot of resources

Which computational infrastructure do you want ?

## Elements of a computational infrastructure

- Hardware
  - The basic bricks
- Software
  - To make hardware usable
- People
  - installers/sys adm. /planners/ users etc..
- Problems to be solved
  - Any action in building such an infrastructure should be motivated by real needs

## Which paradigm/infrastructure for your problem ?

- HPC infrastructure:
  - Hpc systems + high performance network to link them together
- Grid Computing infrastructures :
  - many systems tightly coupled by software, perhaps geographically distributed, to work together on single problems or on related problems

Not an “either/or” question
- Each addresses different needs
- Each are part of an integrated solution

## Which HPC/GRID infrastructure do I need ?

- Which applications ?
  - Parallel
    - Tightly coupled
    - Loosely coupled
    - Embarrassingly
  - Serial
    - Memory /  I/O requirements
- Budget  considerations
- Time to solution considerations

## Summing up

- Modern Science requires a large amount of computing resources  and extended collaboration
- GRID computing address this requirement envision transparent access to resources and dynamic virtual organization interacting space
- HPC and GRID computing are not mutually exclusive but can be both used to address computational resources in a transparent way.