# Overview of Grid middleware concepts

Peter Kacsuk

•MTA SZTAKI, Hungary

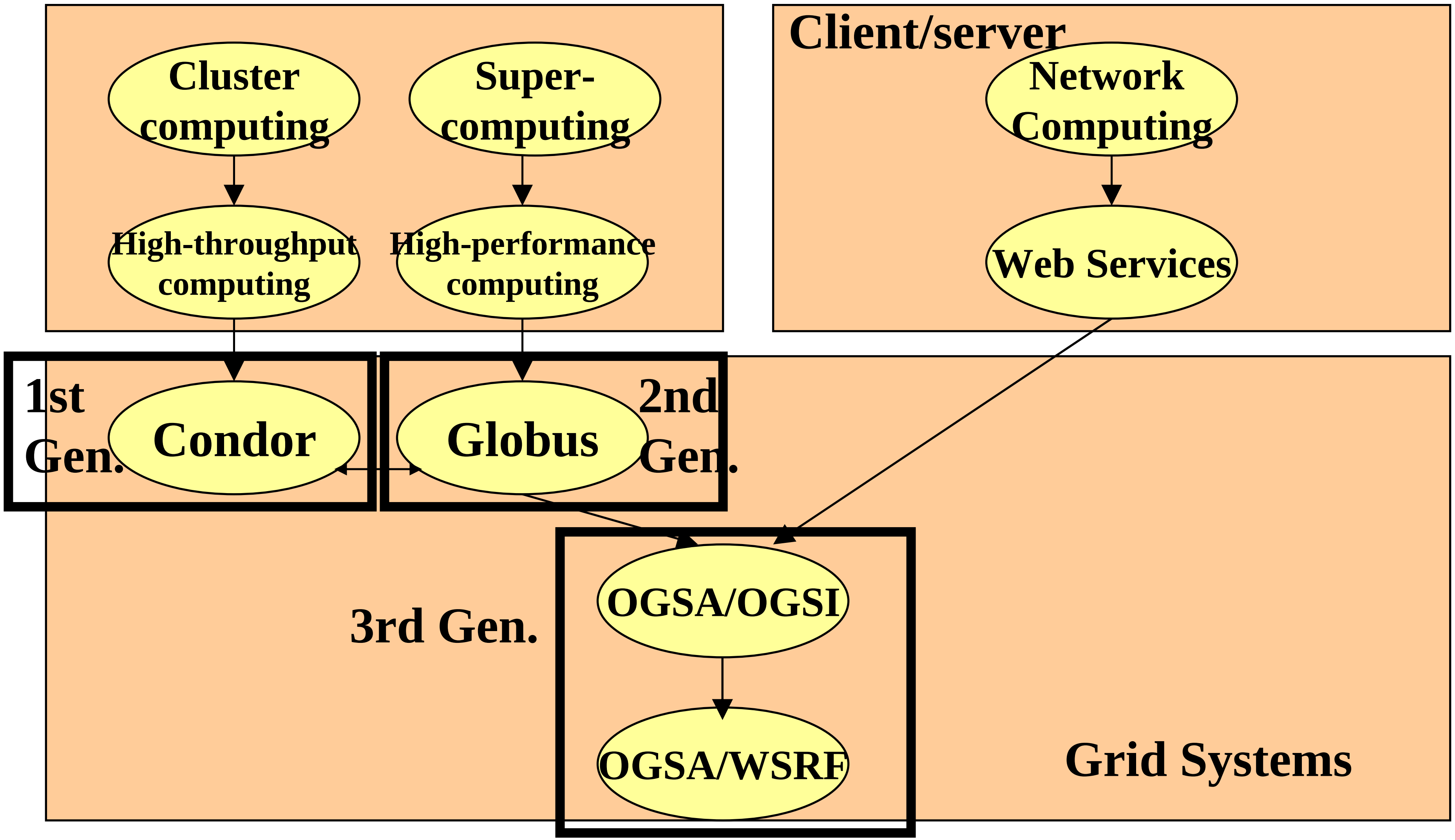
• Univ. Westminster, UK

kacsuk@sztaki.hu

# The goal of this lecture

- To overview the main trends of the fast evolution of Grid systems
- Explaining the main features of the three generation of Grid systems
  - 1st gen. Grids: **Metacomputers**
  - 2nd gen. Grids: **Resource-oriented Grids**
  - 3rd gen. Grids: **Service-oriented Grids**
- To show how these Grid systems can be handled by the users

# Progress in Grid Systems



# 1st Generation Grids
# Metacomputers

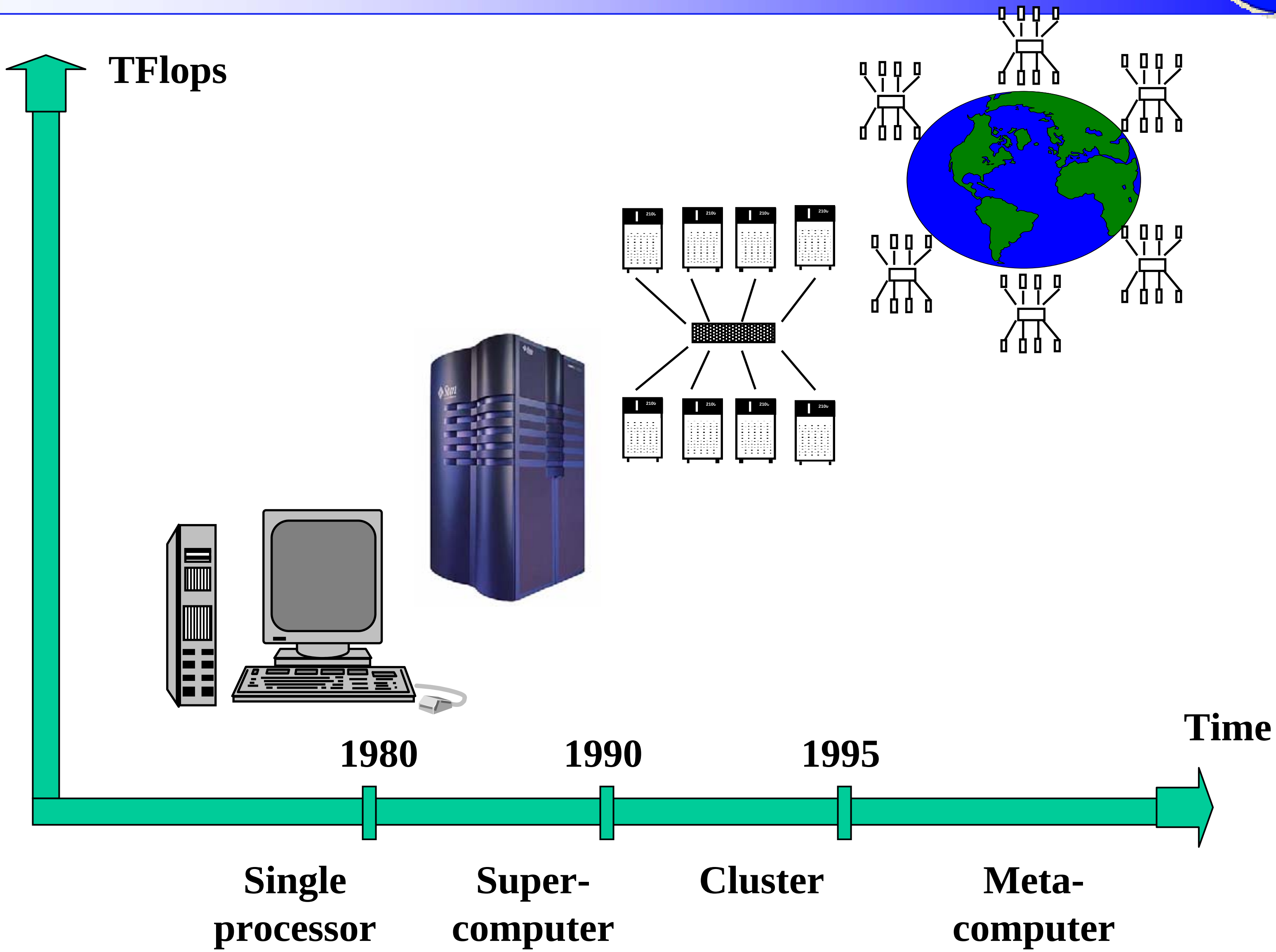
# Original motivation for metacomputing

- **Grand challenge problems** run weeks and months even on supercomputers and clusters

↓

- **Various supercomputers/clusters** must be connected by wide area networks in order to solve grand challenge problems in reasonable time

# Progress to Metacomputers



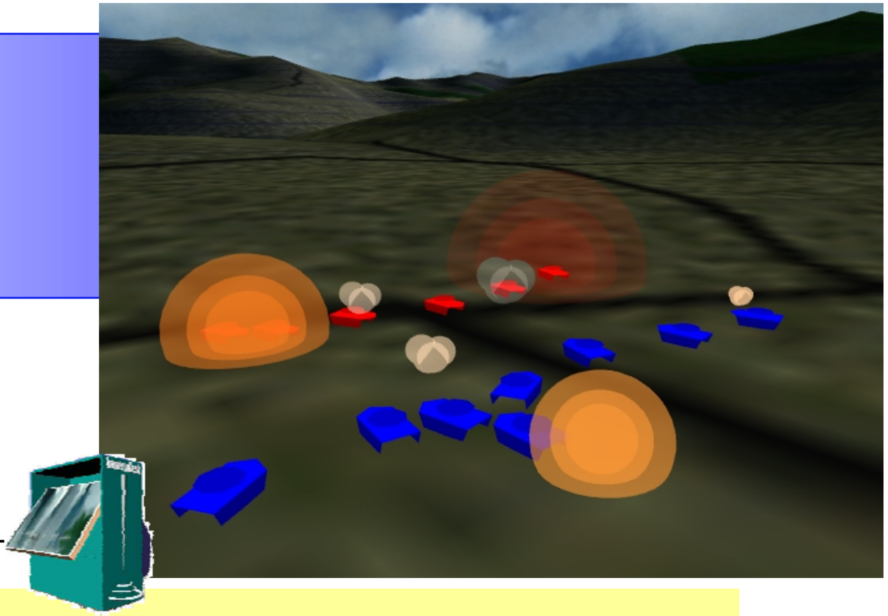
# Original meaning of metacomputing

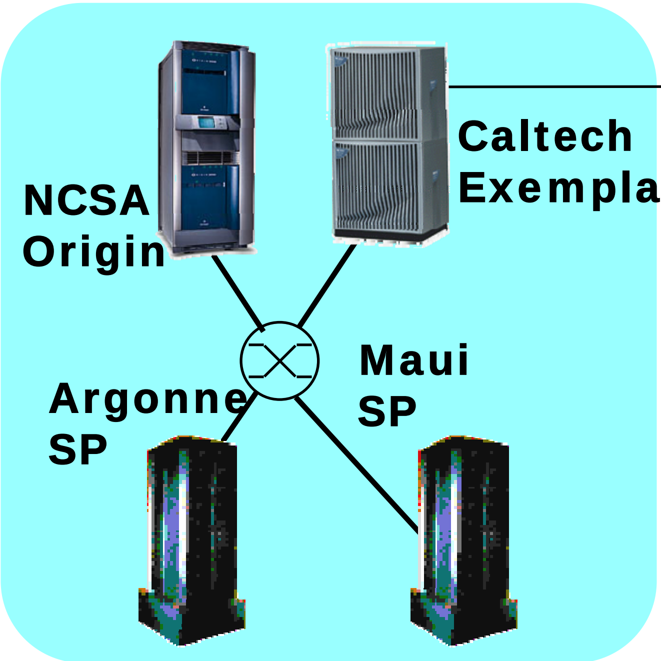
Metacomputing = Super computing + Wide area network

# Original goal of metacomputing:

- **Distributed supercomputing** to achieve **higher performance** than individual supercomputers/clusters can provide

# Distributed Supercomputing






- Issues:
  - Resource discovery, scheduling
  - Configuration
  - Multiple comm methods
  - Message passing (MPI)
  - Scalability
  - Fault tolerance

SF-Express Distributed Interactive Simulation (SC'1995)

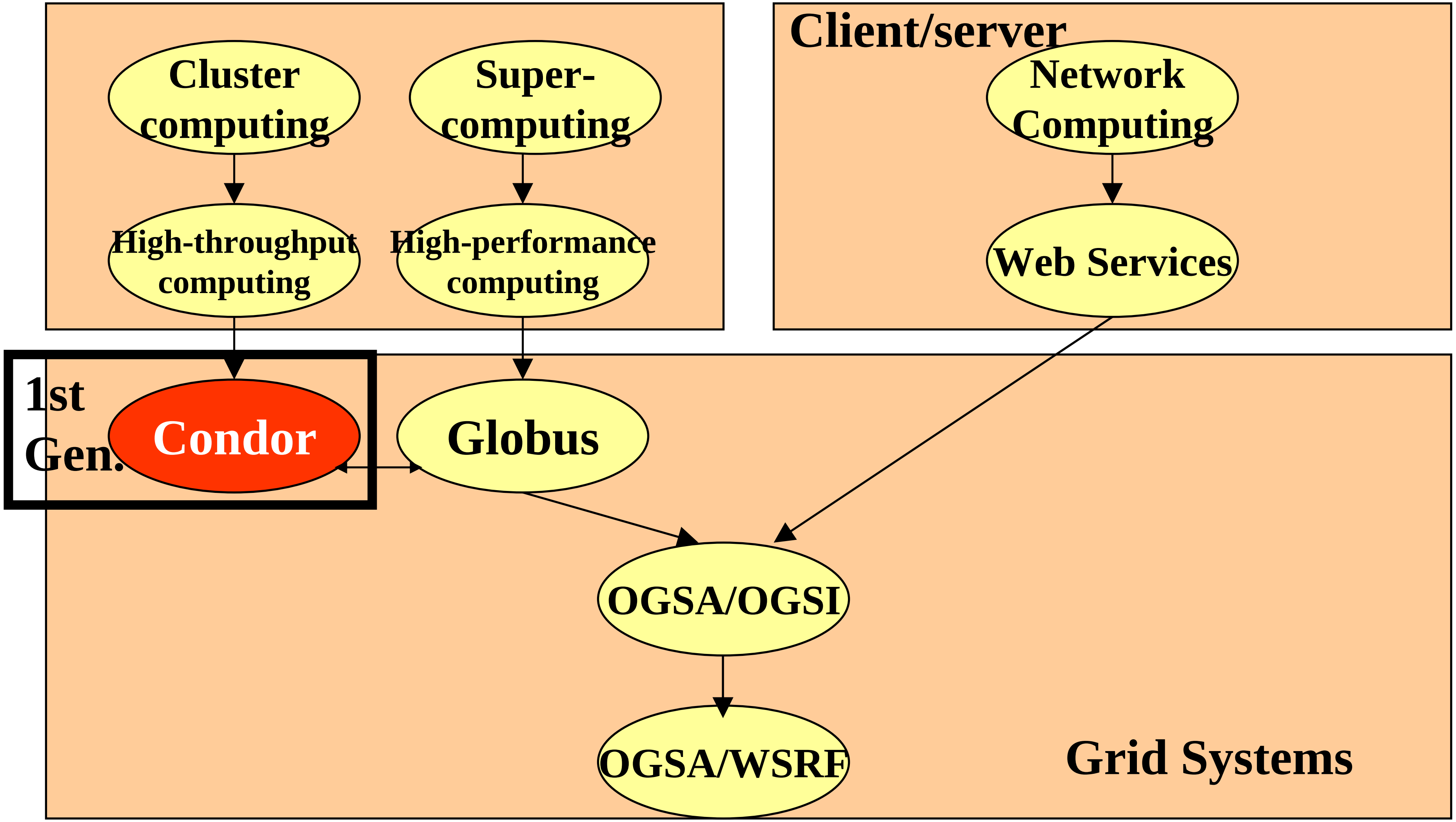
# High-throughput computing (HTC) and the Grid

- **Better usage of** computing and other resources accessible via wide area network

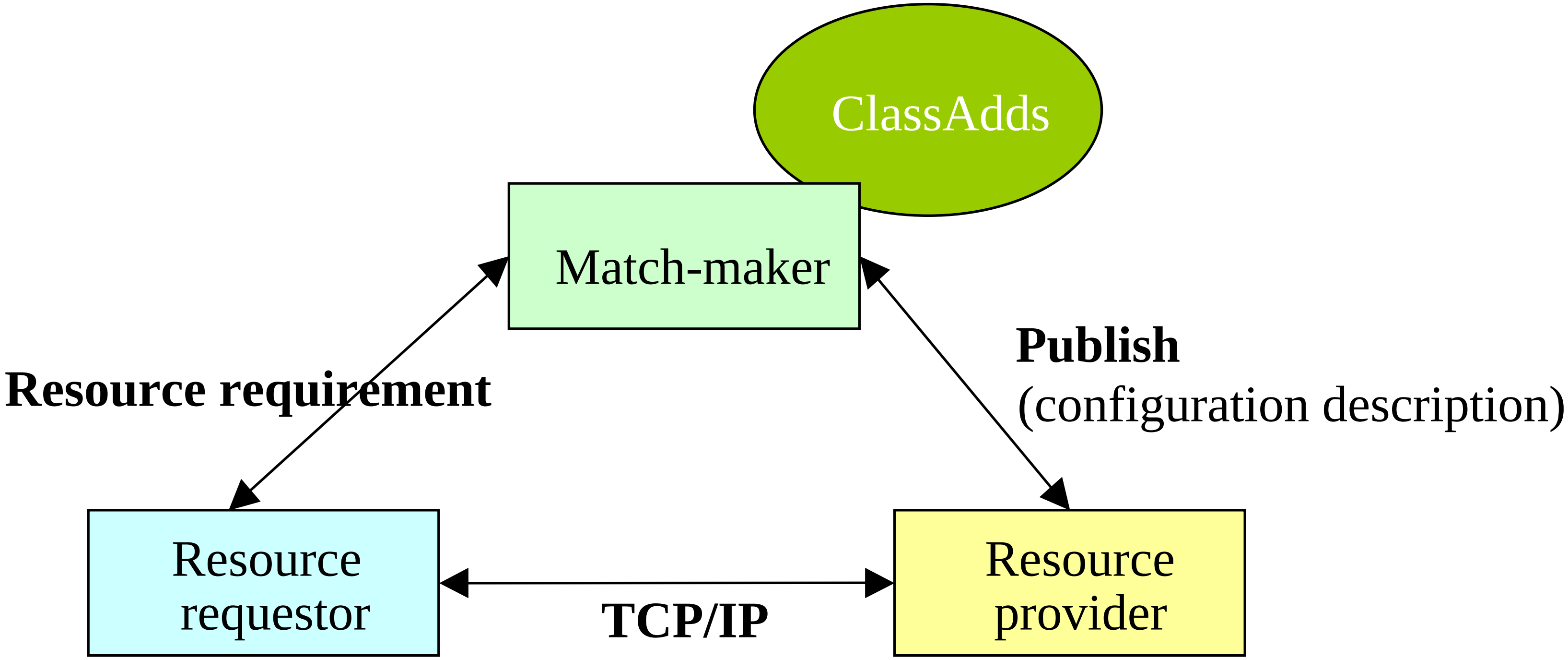
↓

- **To exploit** the **spare cycles of various computers** connected by wide area networks
- **Two main representatives**
  - **SETI**
  - **Condor**

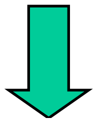
# Progress in Grid Systems



# The Condor model




**Client program moves to resource(s)**

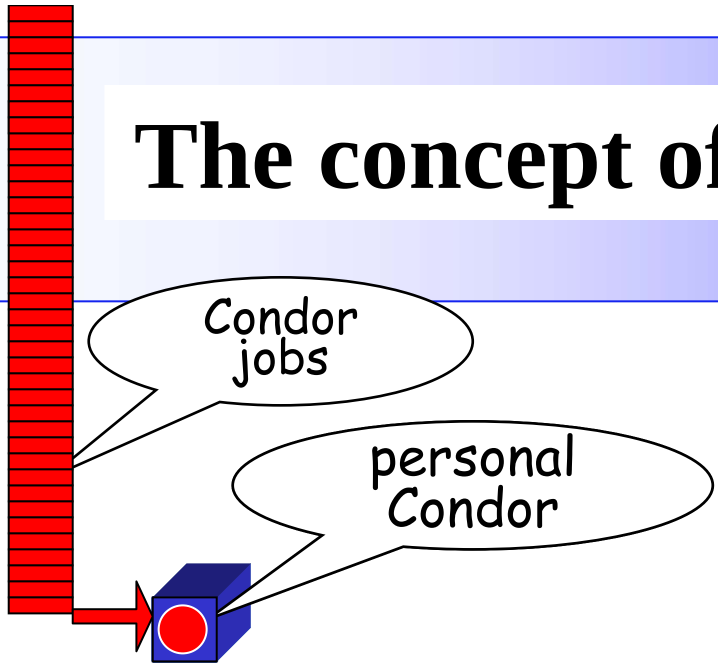


**Security is a serious problem!**

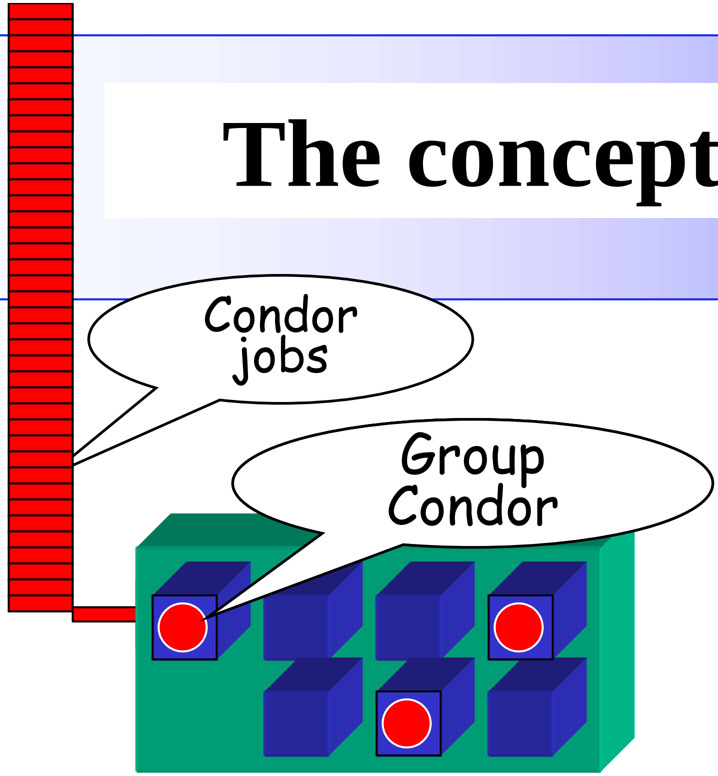
# ClassAds

- Resources of the Grid have different properties (architecture, OS, performance, etc.) and these are described as advertisements (ClassAds)
- Creating a job, we can describe our requirements (and preferencies) for these properties.
- Condor tries to match the requirements and the ClassAds to provide the most optimal resources for our jobs.

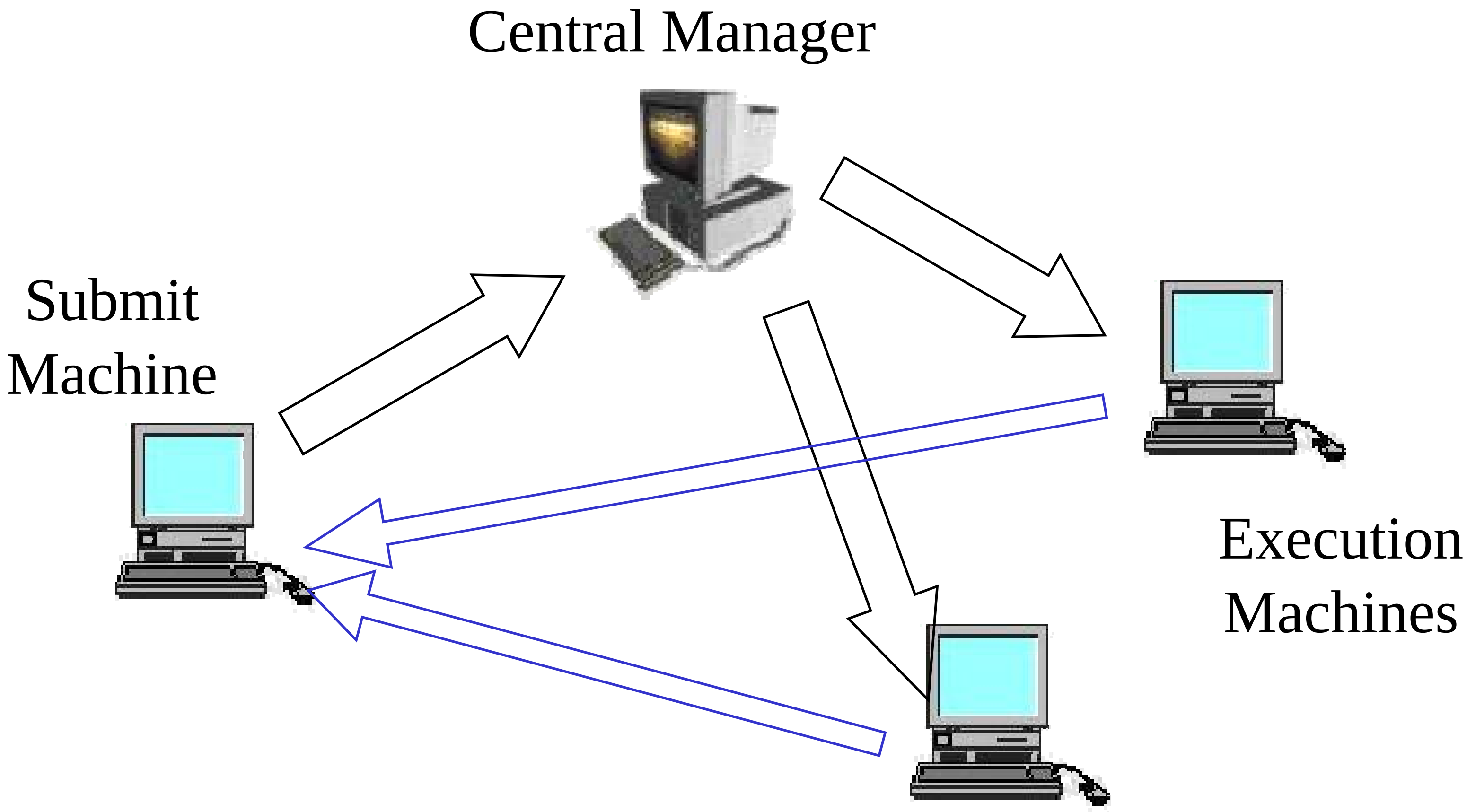
# The concept of personal Condor



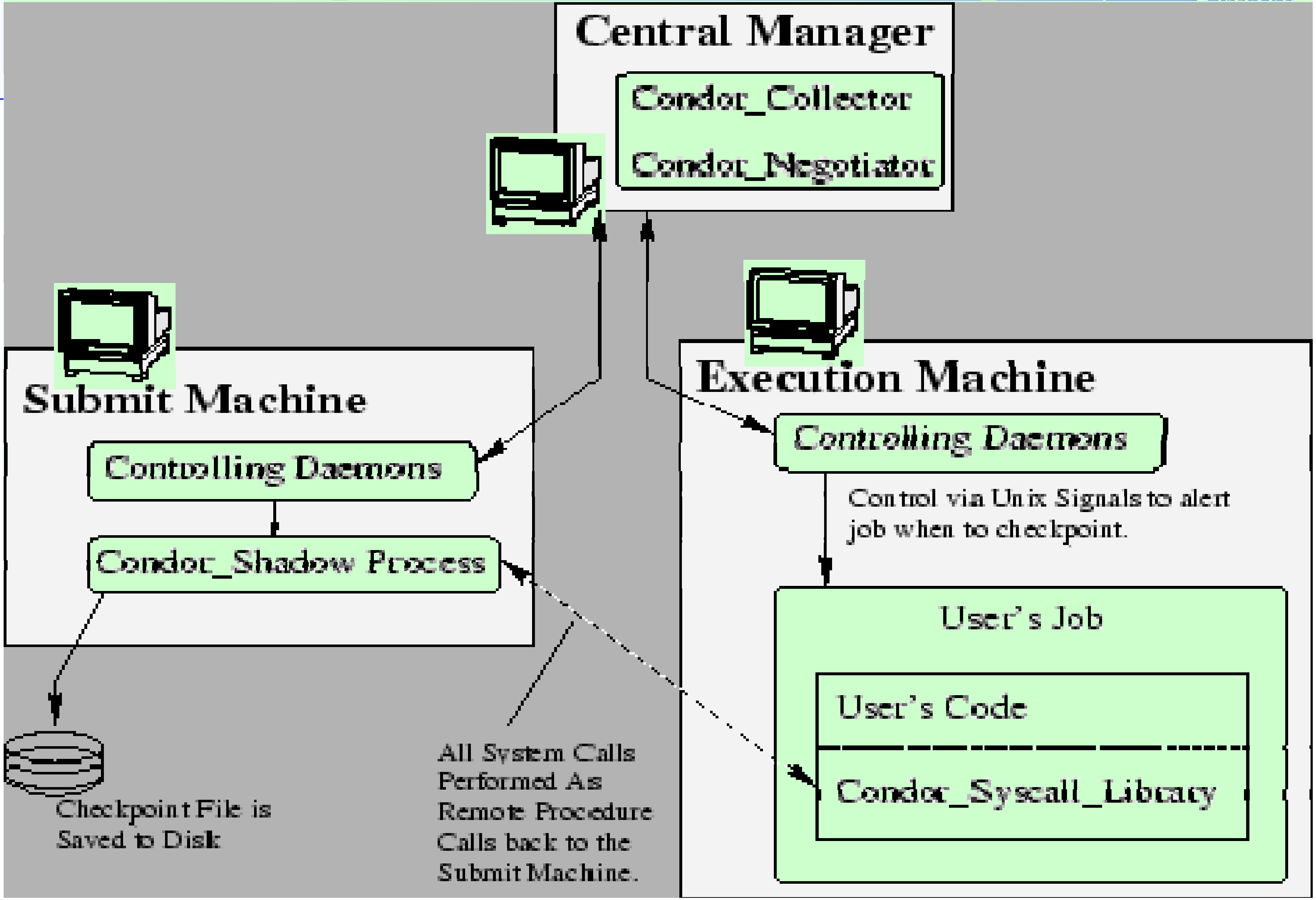
# The concept of Condor pool



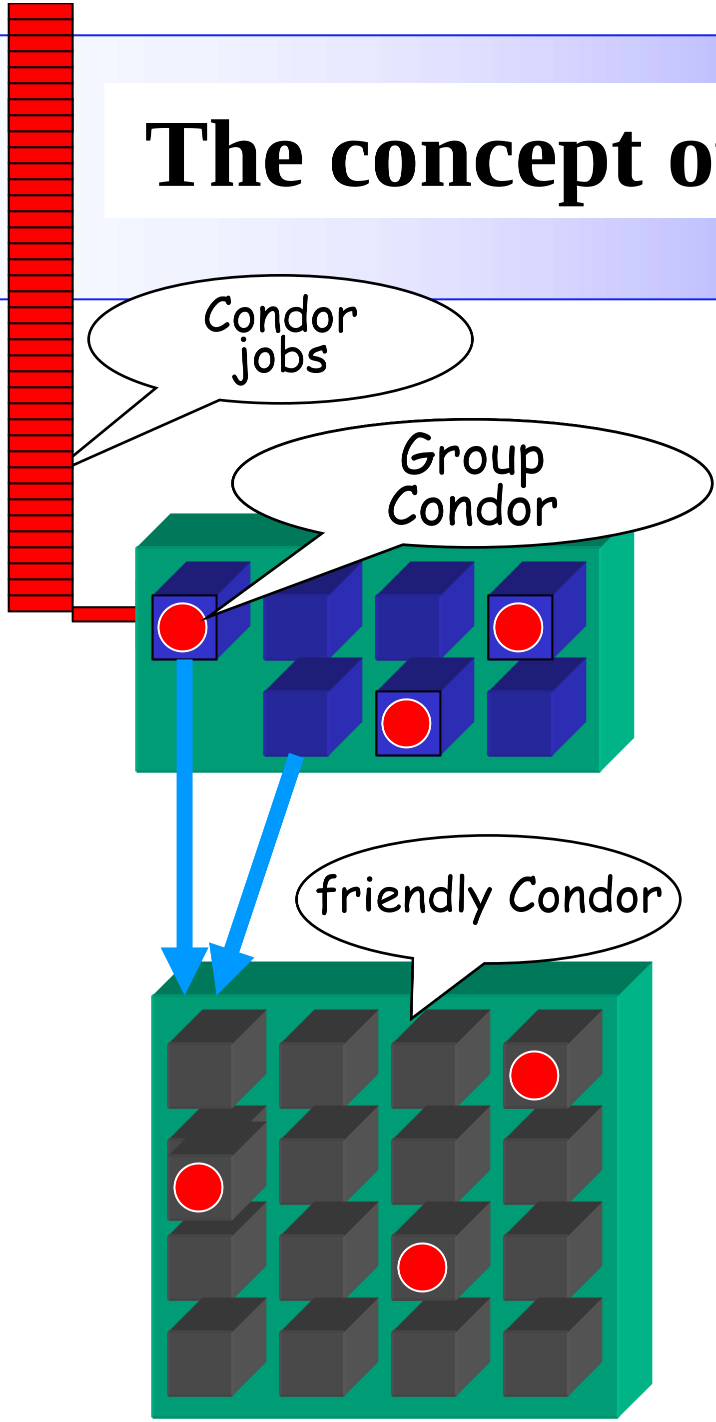
# Architecture of a Condor pool



# Components of a Condor pool

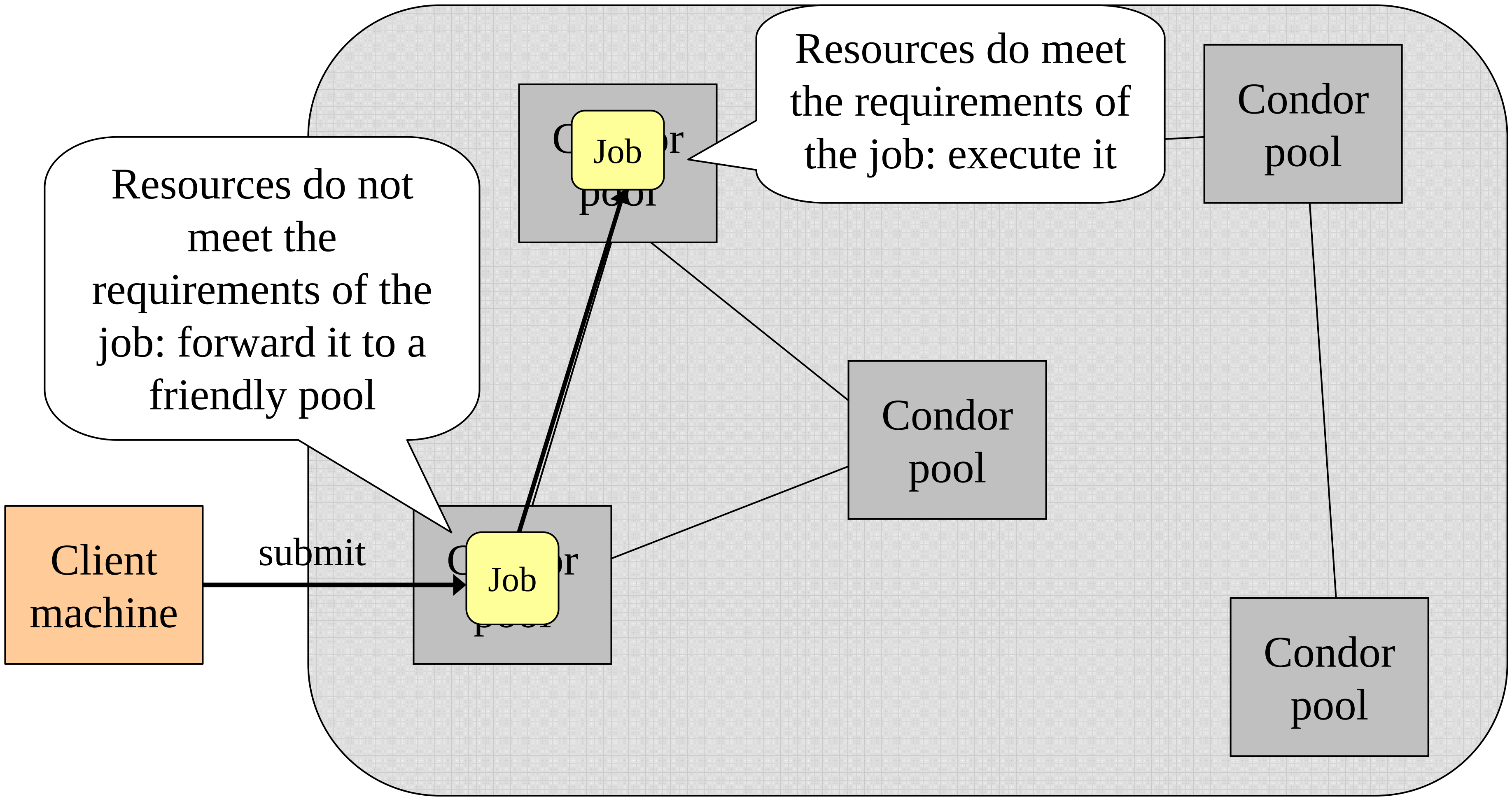


# The concept of Condor flocking

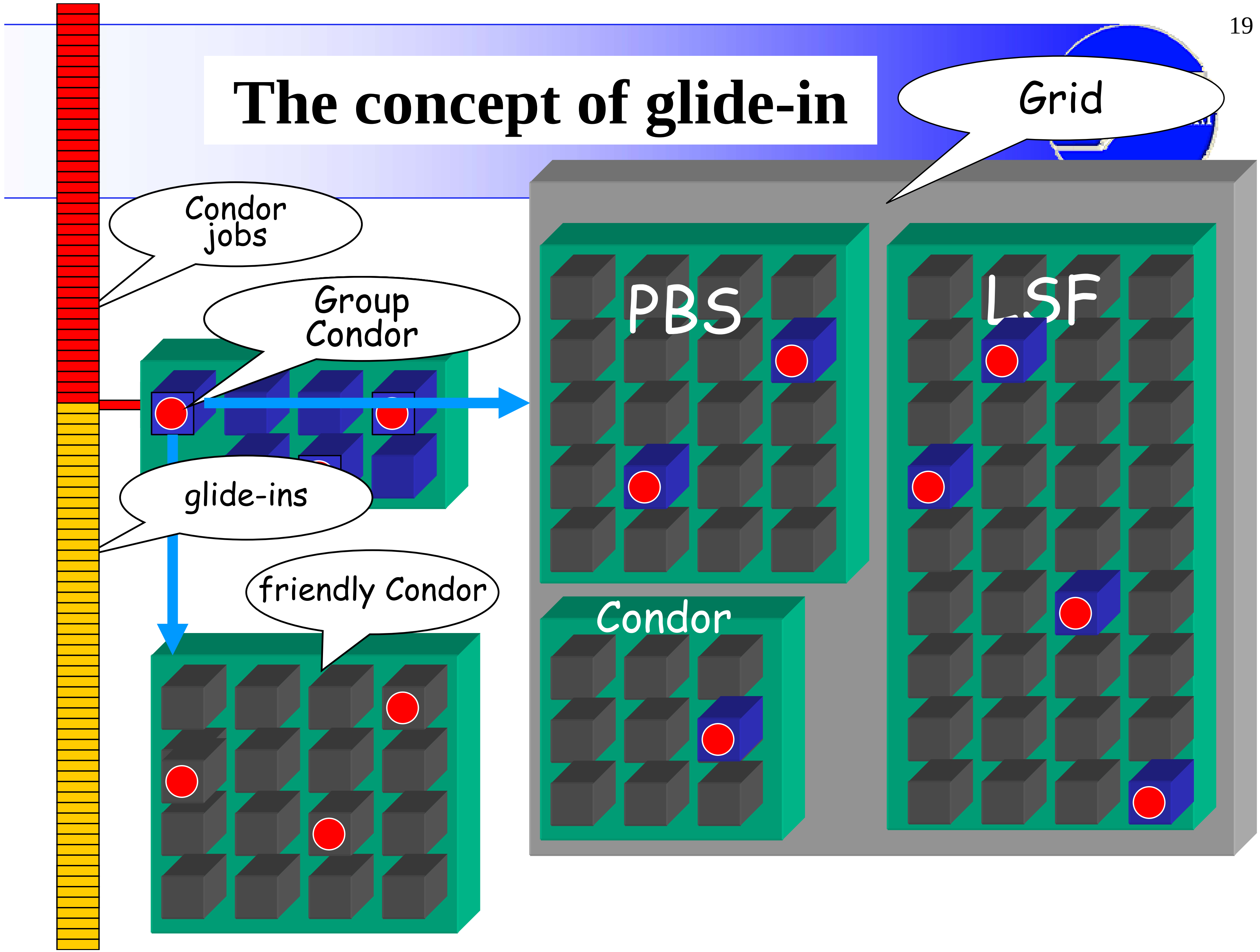



**Your schedd daemons see the CM of the other pool as if it was part of your pool**

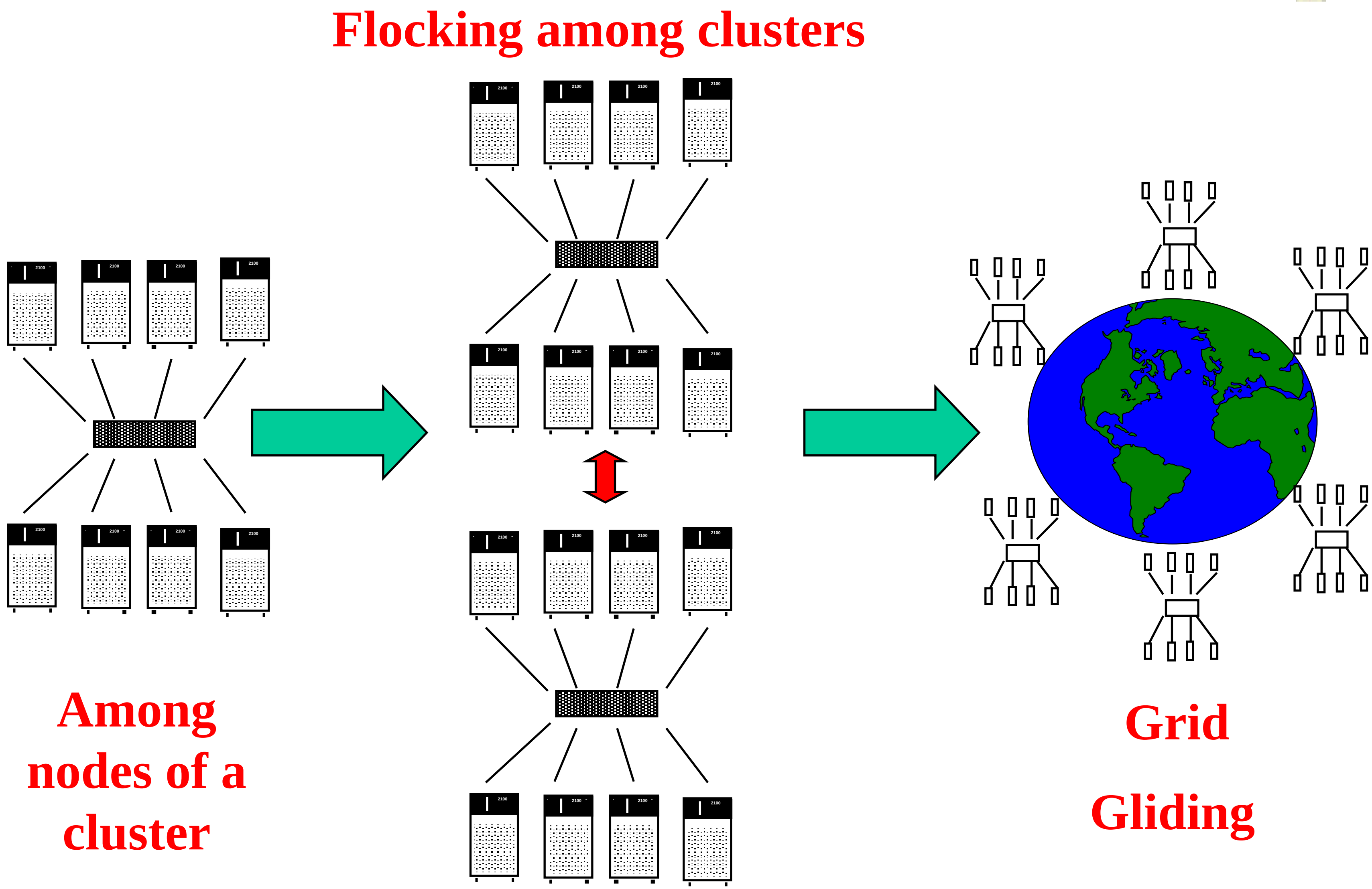
# Condor flocking “grids”



# The concept of glide-in

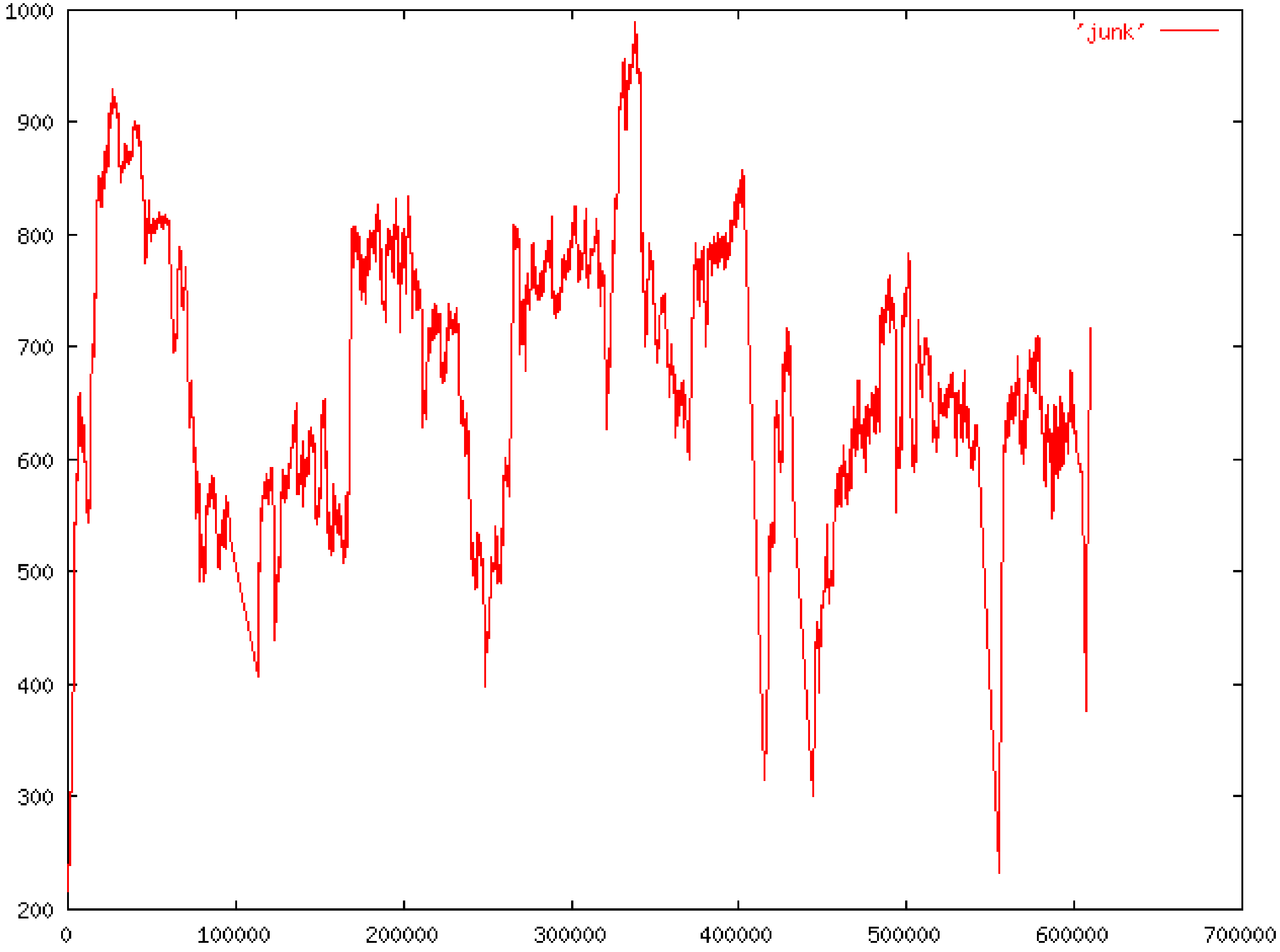


# Three levels of scalability in Condor



# NUG30 - Solved!!!

Number of workers




Solved in 7 days instead of 10.9 years

# NUG30 Personal Grid …

MTA SZTAKI

## Managed by **one** Linux box at Wisconsin

**Flocking:**
-- Condor pool at Wisconsin (500 processors)
-- Condor pool at Georgia Tech (284 Linux boxes)
-- Condor pool at UNM (40 processors)
-- Condor pool at Columbia (16 processors)
-- Condor pool at Northwestern (12 processors)
-- Condor pool at NCSA (65 processors)
-- Condor pool at INFN Italy (54 processors)

**Glide-in:**
-- Origin 2000 (through LSF ) at NCSA. (512 processors)
-- Origin 2000 (through LSF) at Argonne (96 processors)

# Problems with Condor flocking "grids"

- Friendly relationships are defined **statically**.
- **Firewalls are not allowed** between friendly pools.
- Client can not choose resources (pools) directly.
- Private (non-standard) "**Condor protocols**" are used to connect friendly pools together.
- **Not service-oriented**

# 2nd Generation Grids
# Resource-oriented Grid

# The main goal of 2nd gen. Grids

- To enable a
  - geographically distributed community [of thousands]
  - to perform sophisticated, computationally intensive analyses
  - on large set **(Petabytes) of data**
- To provide
  - on demand
  - **dynamic resource aggregation**
  - as virtual organizations

Example virtual organizations :

- Physics community (EDG, EGEE)
- Climate community, etc.

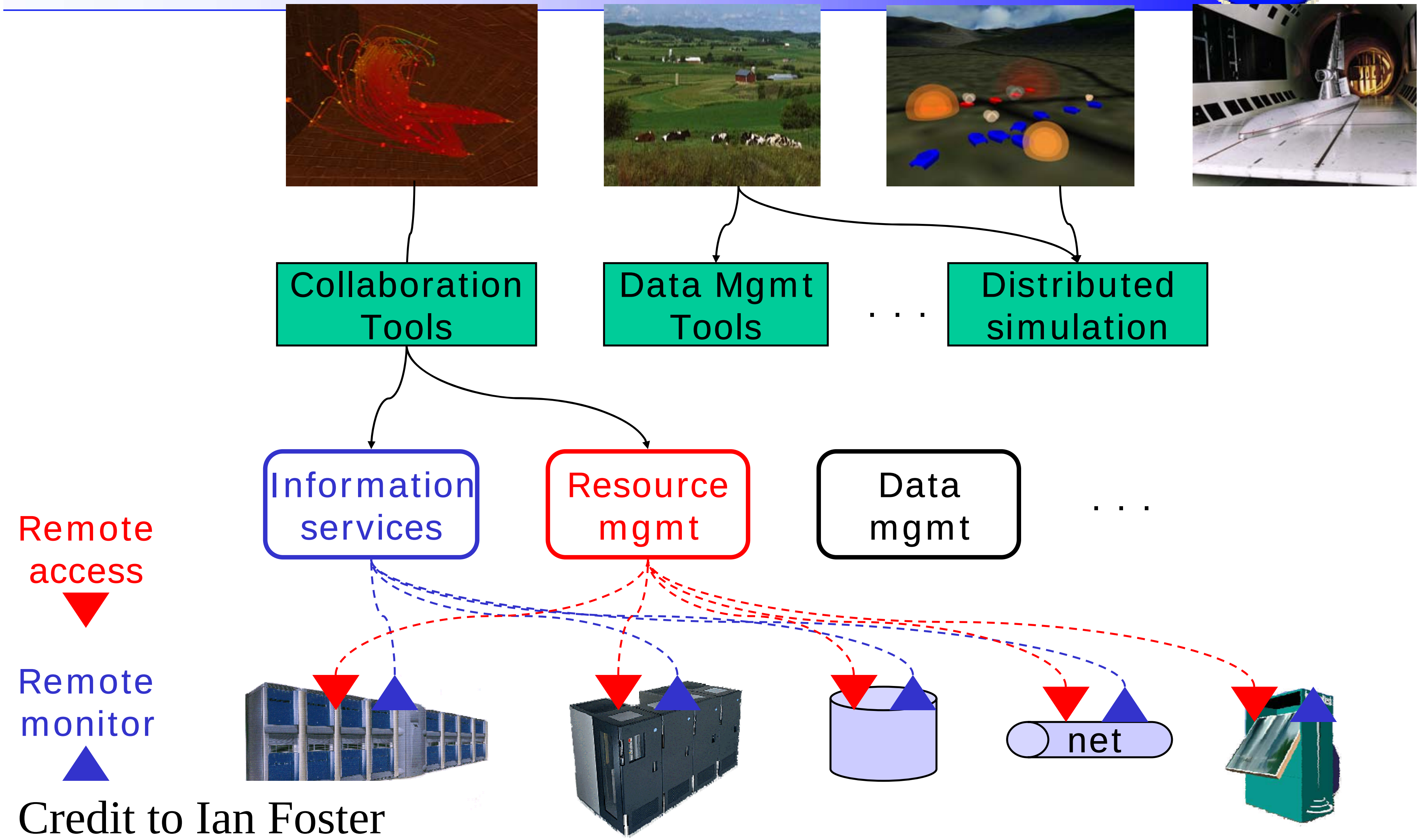
# Resource intensive issues include

- Harness data, storage, computing and network resources located in distinct administrative domains
- Respect local and global policies governing what can be used for what
- Schedule resources efficiently, again subject to local and global constraints
- Achieve high performance, with respect to both speed and reliability

# Grid Protocols, Services and Tools

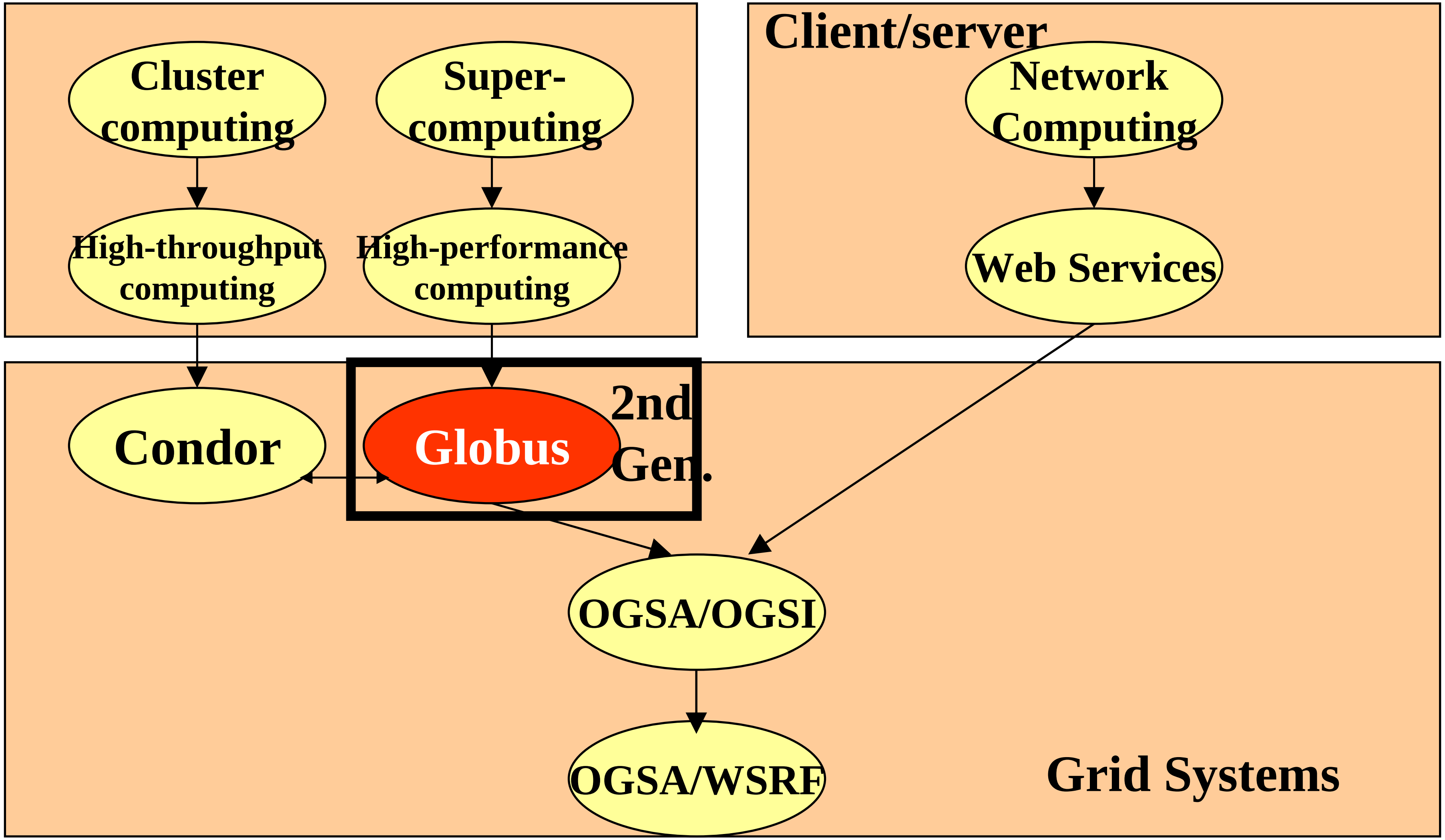
- **Protocol**-based access to resources
  - Mask local heterogeneities
  - Negotiate multi-domain security, policy
  - “Grid-enabled” resources speak Grid protocols
  - Multiple implementations are possible
- Broad deployment of protocols facilitates creation of **services** that provide integrated view of **distributed resources**
- **Tools** use protocols and services to enable specific classes of applications

# The Role of Grid Middleware and Tools




Credit to Ian Foster

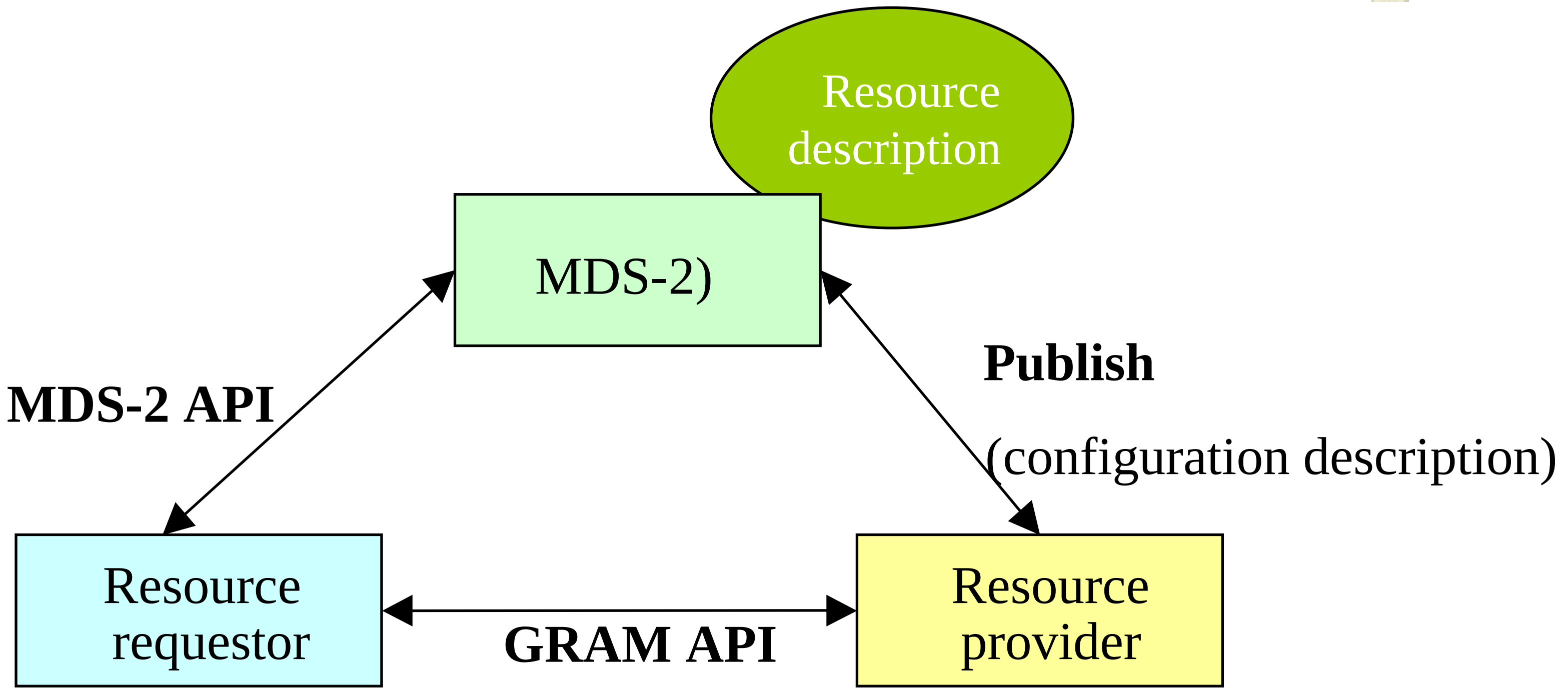
# Progress in Grid Systems



# Solutions by Globus (GT-2)

- Creation of Virtual Organizations (VOs)
- **Standard protocols** are used to connect Globus sites
- Security issues are basically solved
  - **Firewalls are allowed** between Grid sites
  - PKI: CAs and X.509 certificates
  - SSL for authentication and message protection
- The client does not need account on every Globus site:
  - Proxies and delegation for secure **single Sign-on**
- Still:
  - provides metacomputing facilities (MPICH-G2)
  - **Not service-oriented either**

# The Globus-2 model



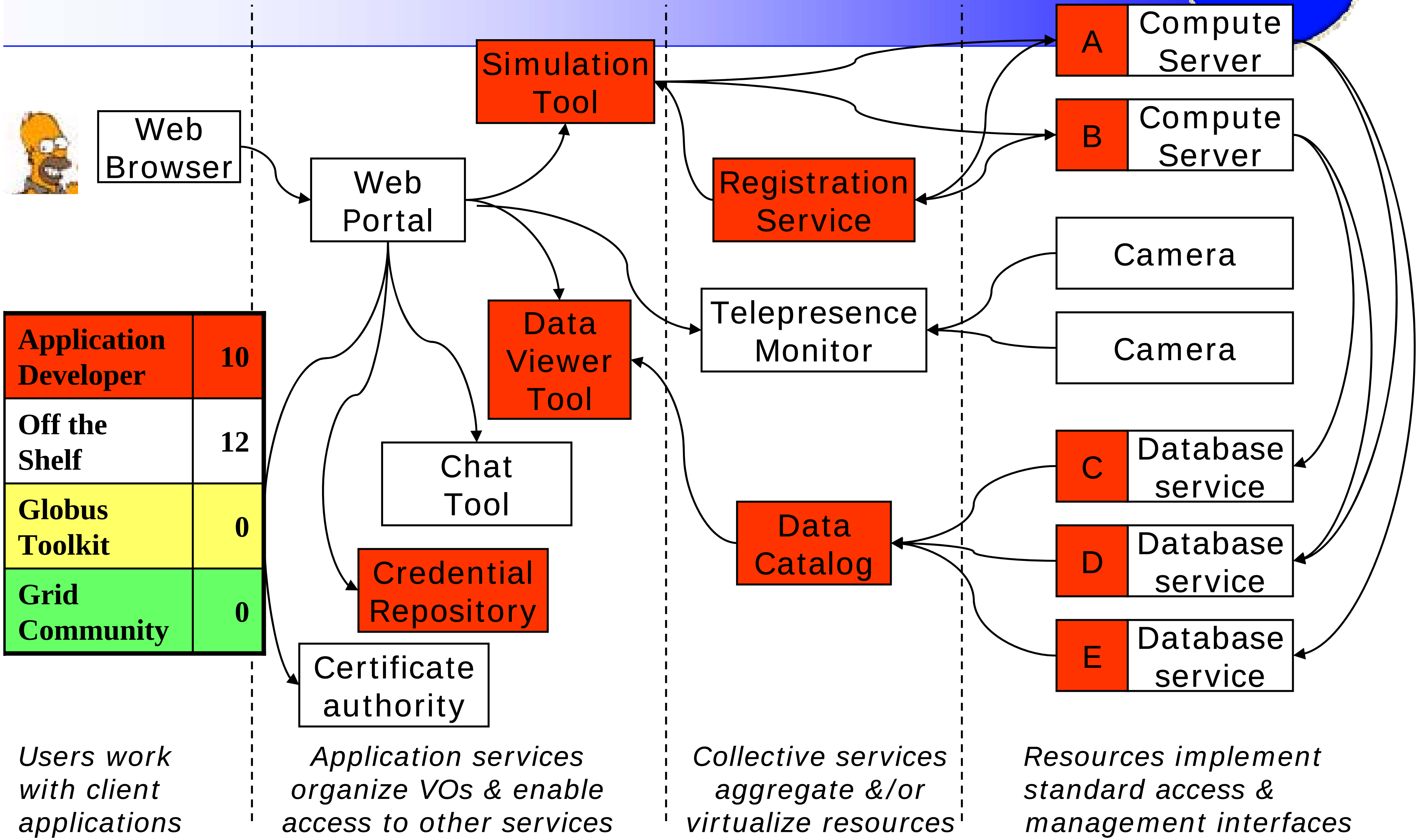

**Client program moves to resource(s)**

**Security is a serious problem!**

# The Role of the Globus Toolkit

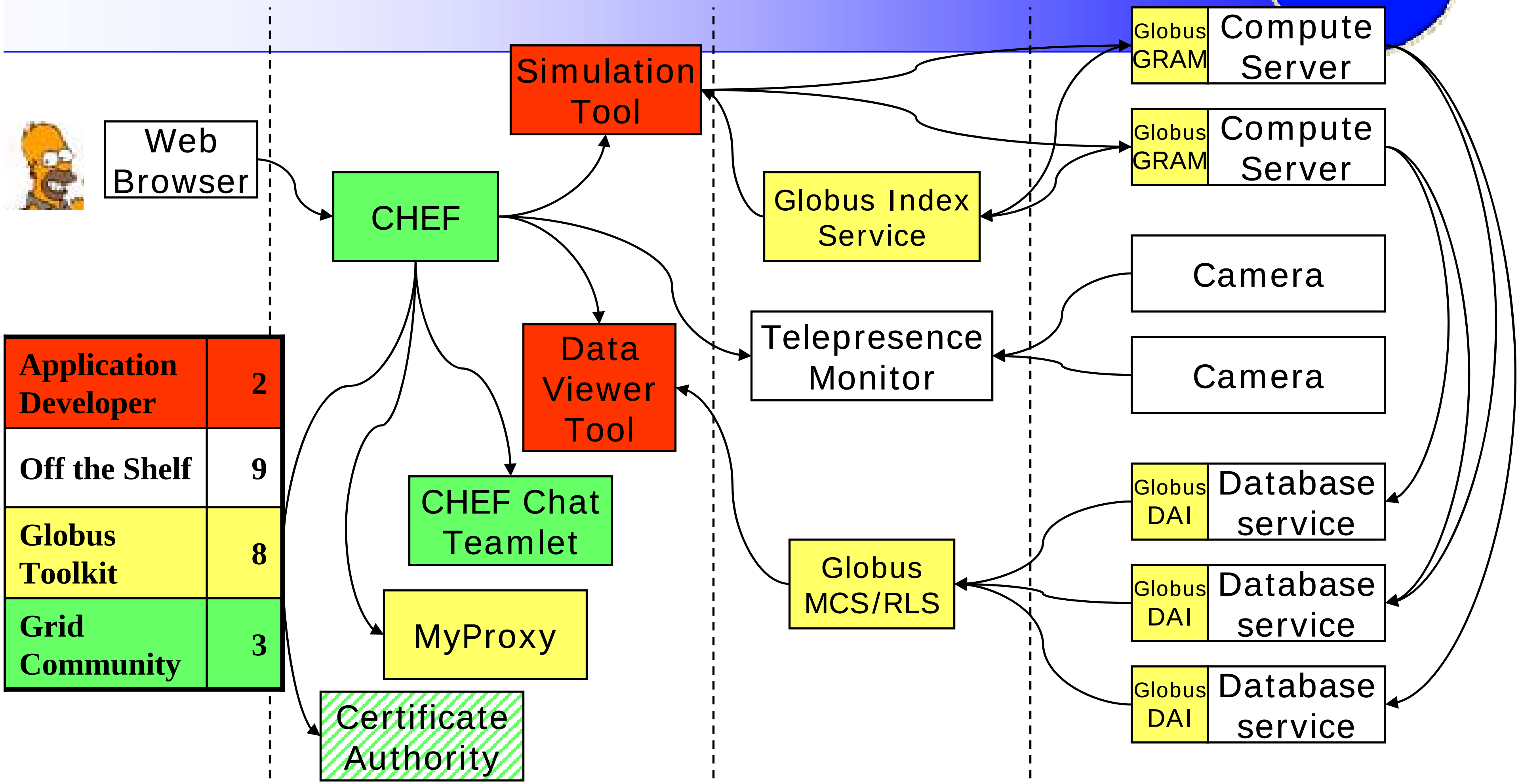
- A collection of solutions to problems that come up frequently when building collaborative distributed applications
- Heterogeneity
  - A focus, in particular, on overcoming heterogeneity for application developers
- Standards
  - We capitalize on and encourage use of existing standards (IETF, W3C, OASIS, GGF)
  - GT also includes reference implementations of new/proposed standards in these organizations

# Without the Globus Toolkit




| | |
|---|---|
| **Application Developer** | **10** |
| **Off the Shelf** | **12** |
| **Globus Toolkit** | **0** |
| **Grid Community** | **0** |

# A possibility with the Globus Toolkit



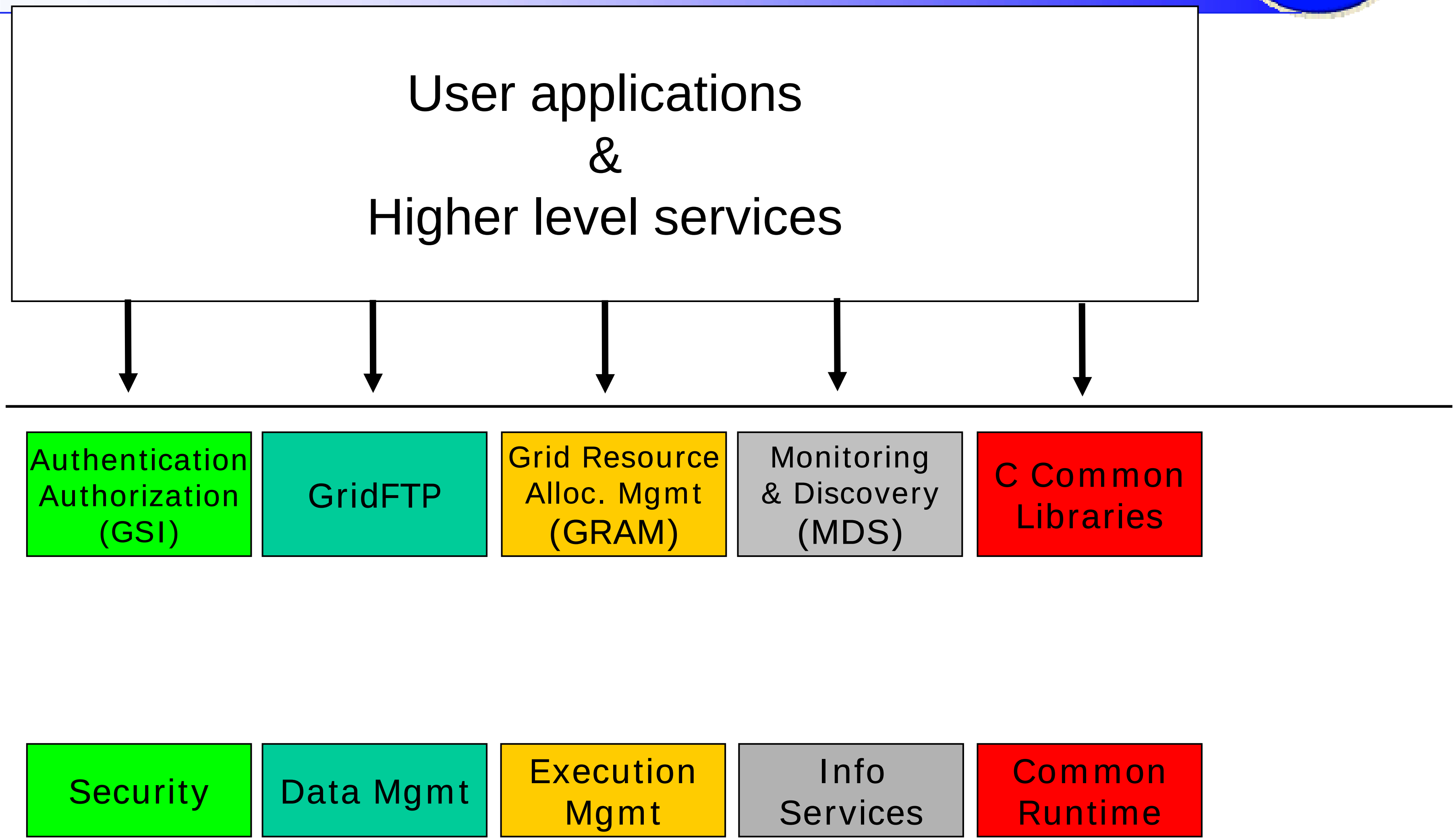

*Users work with client applications*

*Application services organize VOs & enable access to other services*

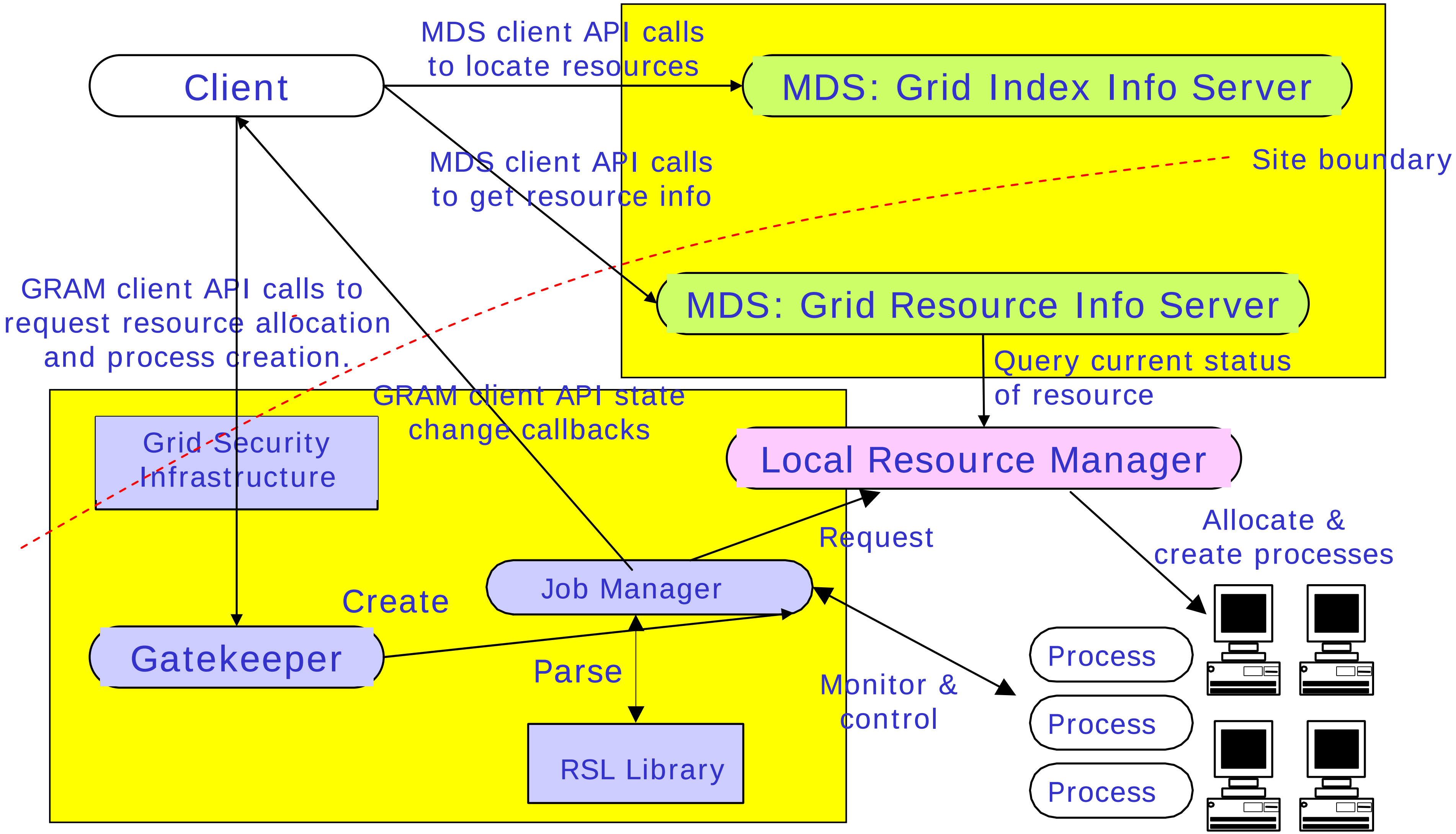
*Collective services aggregate &/or virtualize resources*

*Resources implement standard access & management interfaces*

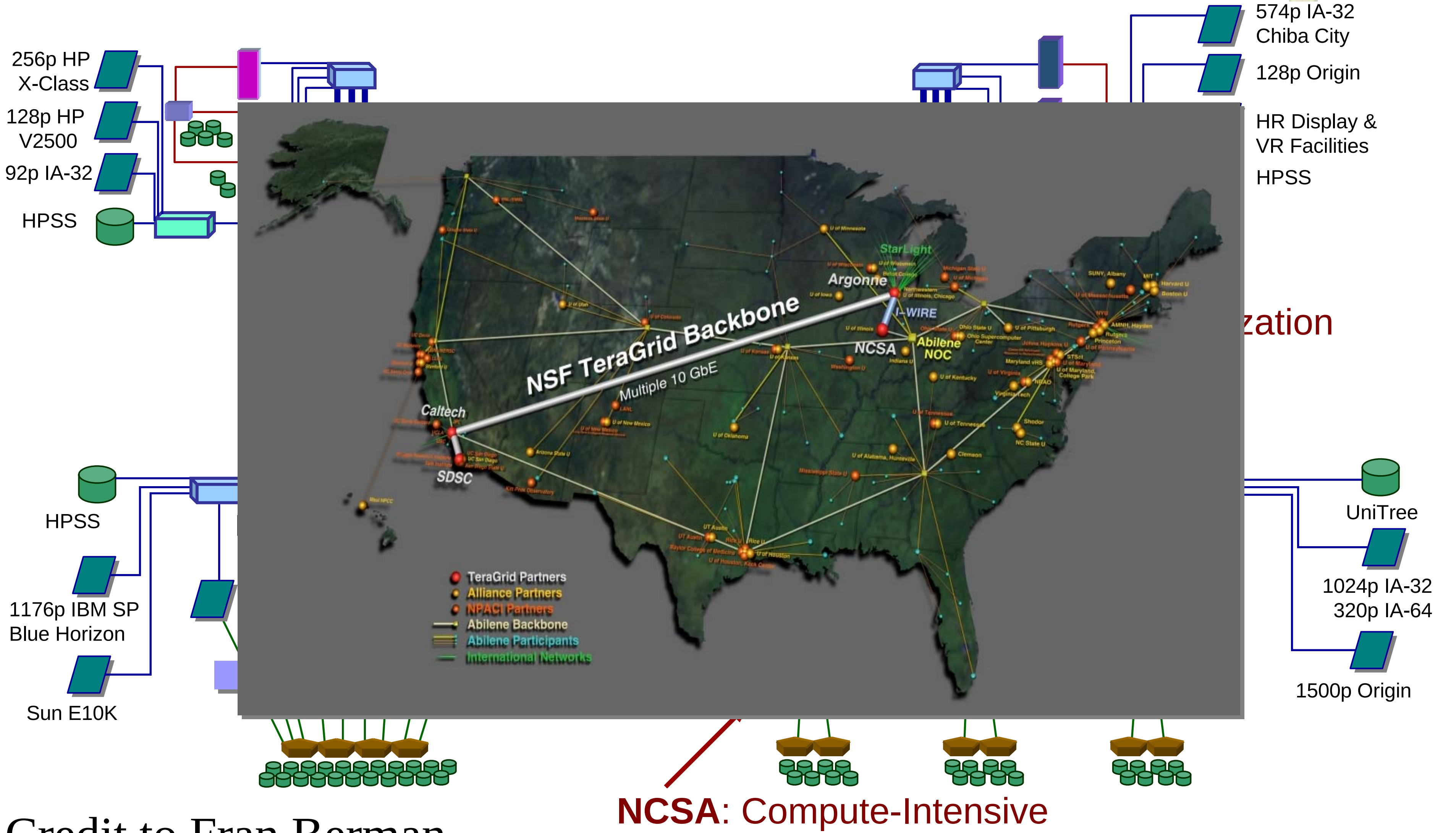
# Globus Toolkit version 2 (GT2)



# Globus Components



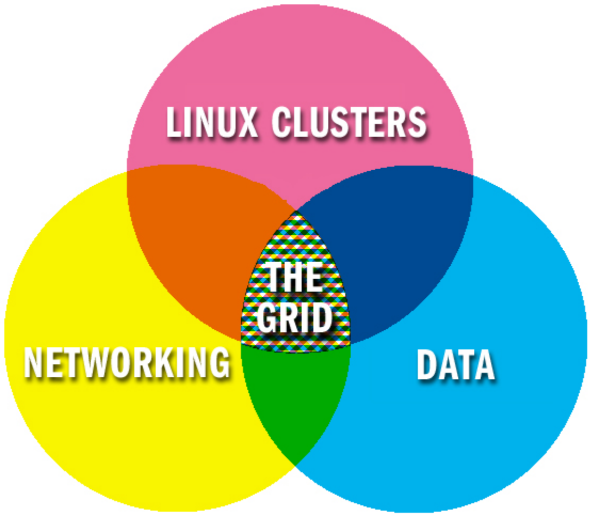
# Example 1 for a GT2 Grid: TeraGrid




Credit to Fran Berman

# TeraGrid Common Infrastructure Environment

- Linux Operating Environment
- Basic and Core Globus Services
  - **GSI (Grid Security Infrastructure)**
  - **GSI-enabled SSH and GSIFTP**
  - **GRAM (Grid Resource Allocation & Management)**
  - **GridFTP**
  - **Information Service**
  - Distributed accounting
  - MPICH-G2
  - Science Portals




- Advanced and Data Services
  - Replica Management Tools
  - GRAM-2 (GRAM extensions)
  - Condor-G (as brokering “super scheduler”)
  - SDSC SRB (Storage Resource Broker)

Credit to Fran Berman

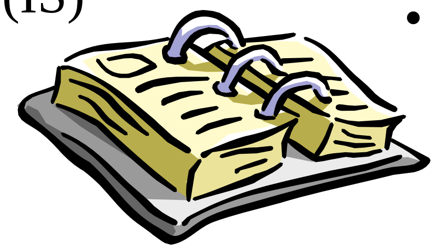
# Example 2 for a GT2 Grid: LHC Grid and LCG-2

- **LHC Grid**
  - A **homogeneous** Grid developed by CERN
  - **Restrictive policies** (global policies overrule local policies)
  - A **dedicated** Grid to the Large Hydron Collider experiments
- **LCG-2**
  - A **homogeneous** Grid developed by CERN and the EDG and EGEE projects
  - **Restrictive policies** (global policies overrule local policies)
  - A **non-dedicated** Grid
  - Works 24 hours/day and has been used in EGEE and EGEE-related Grids (SEEGRID, BalticGrid, etc.)

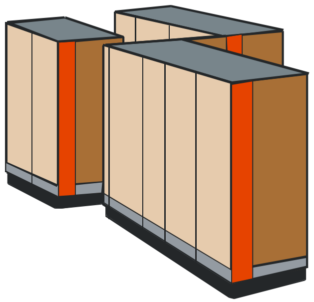
# Main Logical Machine Types (Services) in LCG-2

- User Interface (UI)

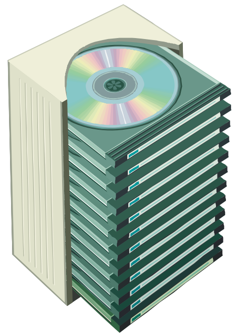
- Information Service (IS)



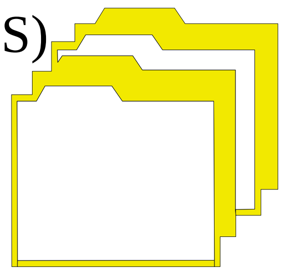
- Computing Element (CE)
  - Frontend Node
  - Worker Nodes (WN)



- Storage Element (SE)

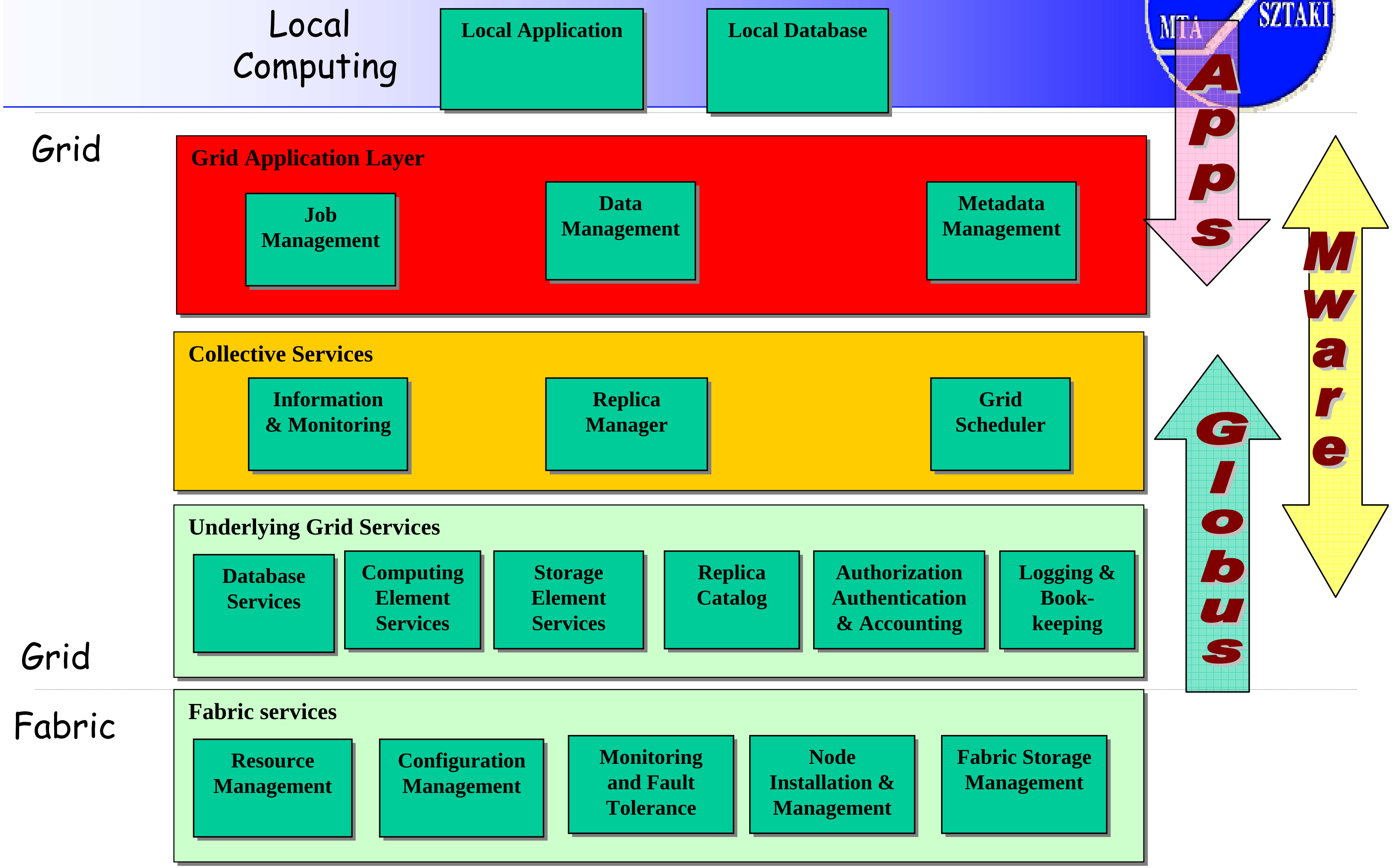


- Replica Catalog (RC,RLS)



Resource Broker (RB)

# The LCG-2 Architecture



# 3rd Generation Grids

**Service-oriented Grids**

**OGSA**

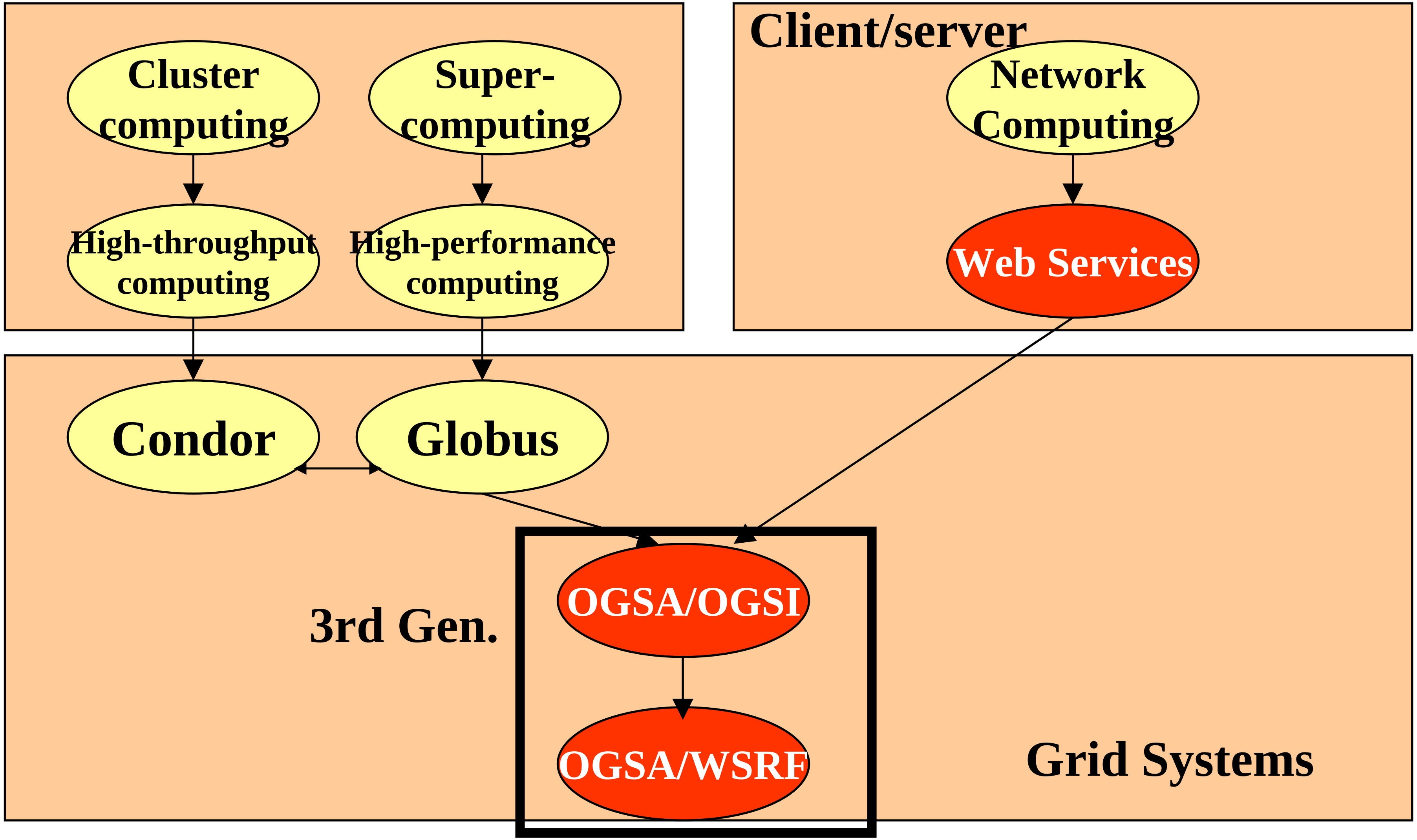
**(Open Grid Service Architecture)**

**and**

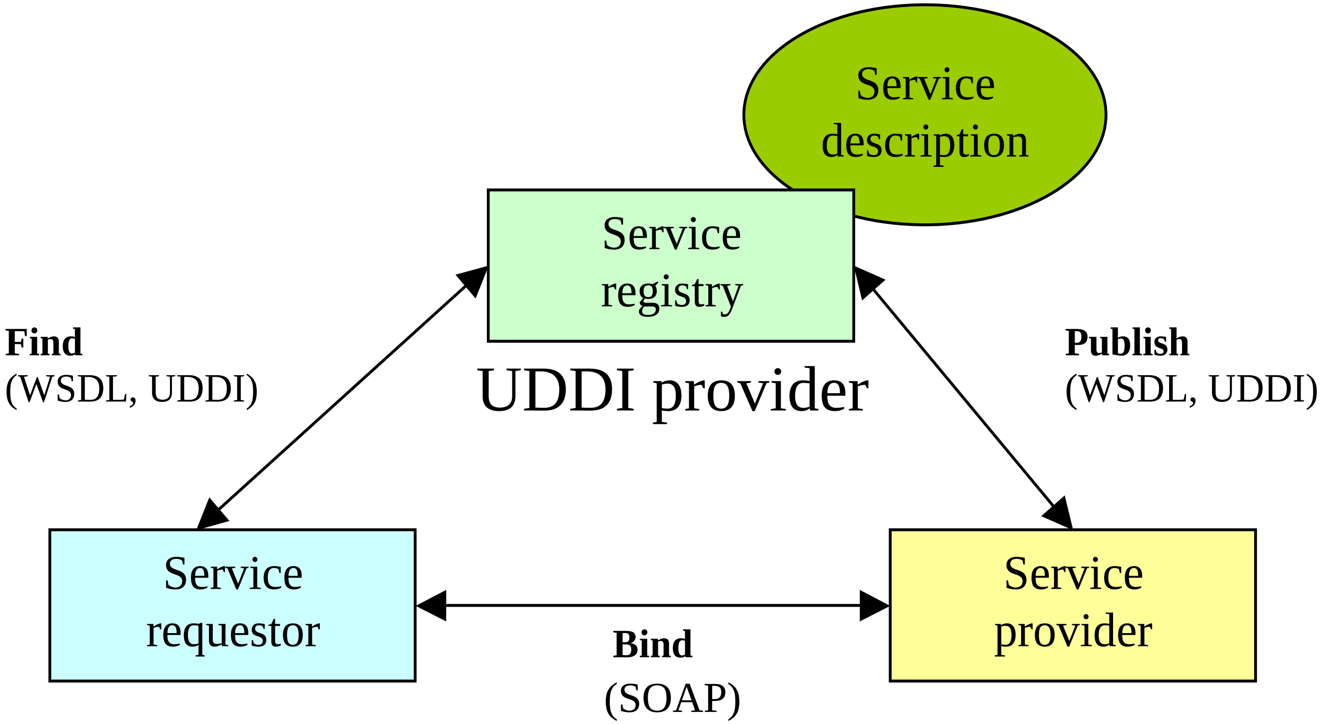
**WSRF**

**(Web Services Resource Framework)**

# Progress in Grid Systems



# The Web Services model

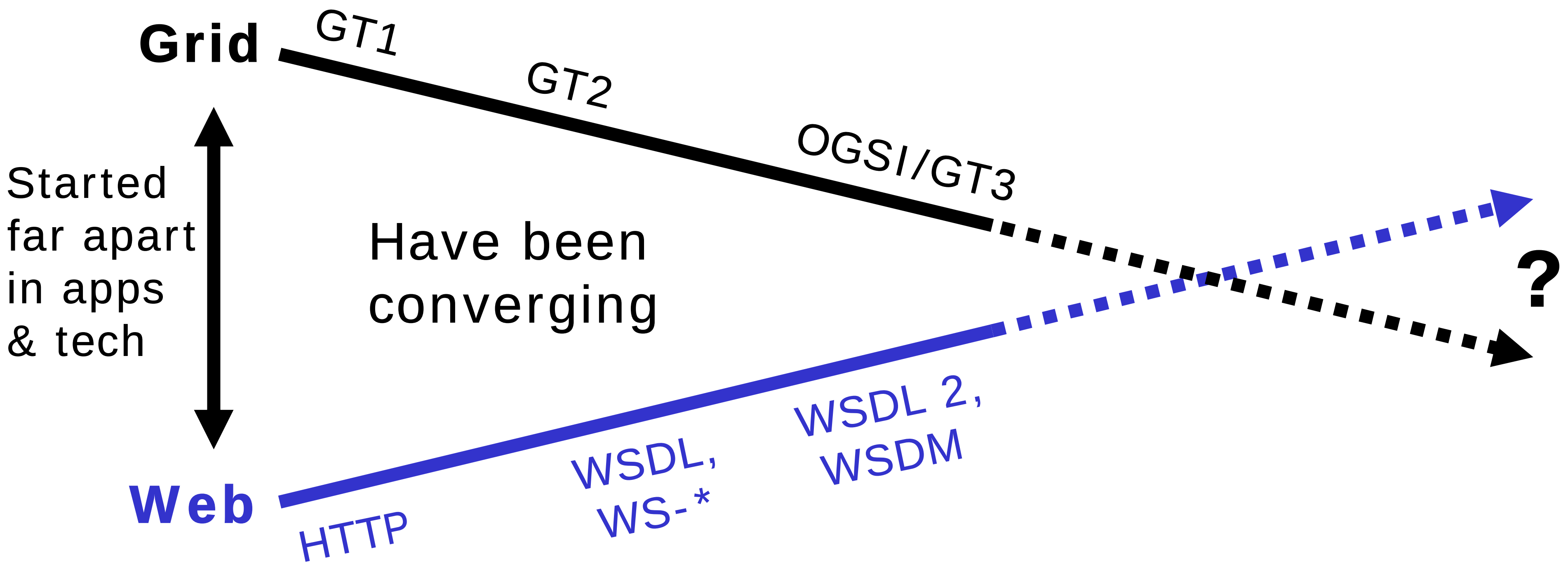



**Predefined programs (services) wait for invocation**

⇓

**Much more secure than the GT-2 concept**

# Grid and Web Services: Convergence

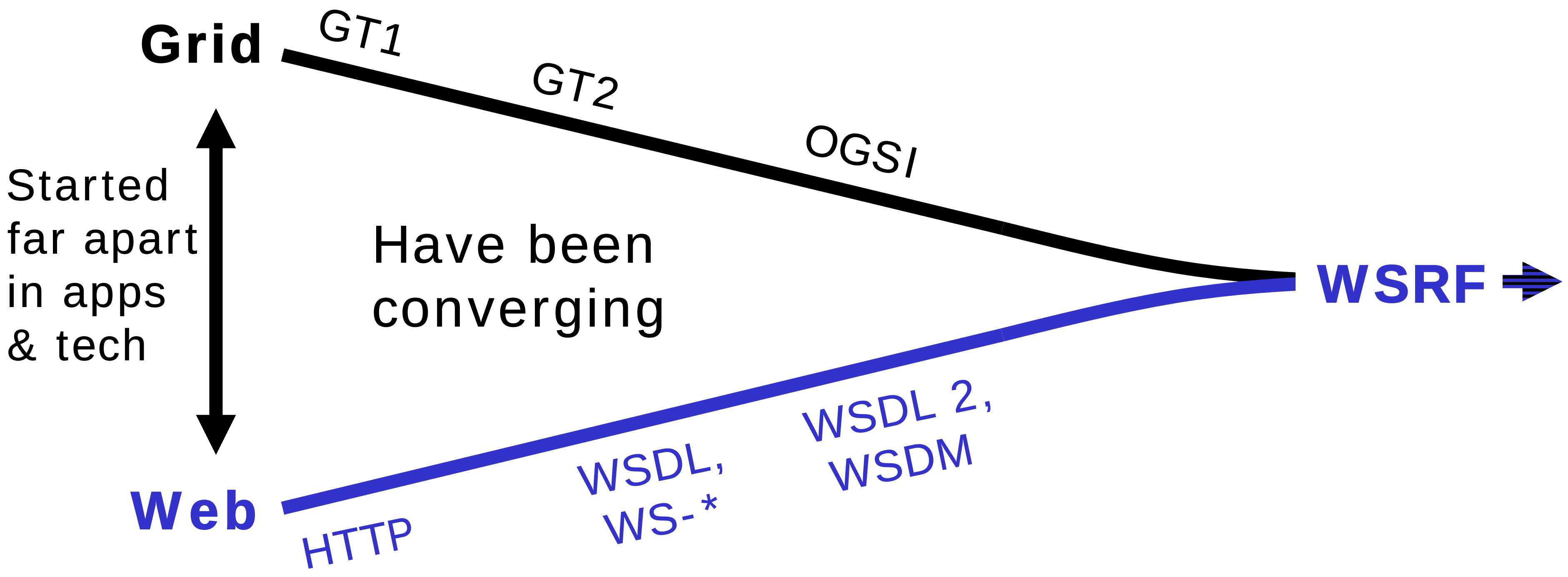



However, despite enthusiasm for OGSI, adoption within Web community turned out to be problematic

# Concerns

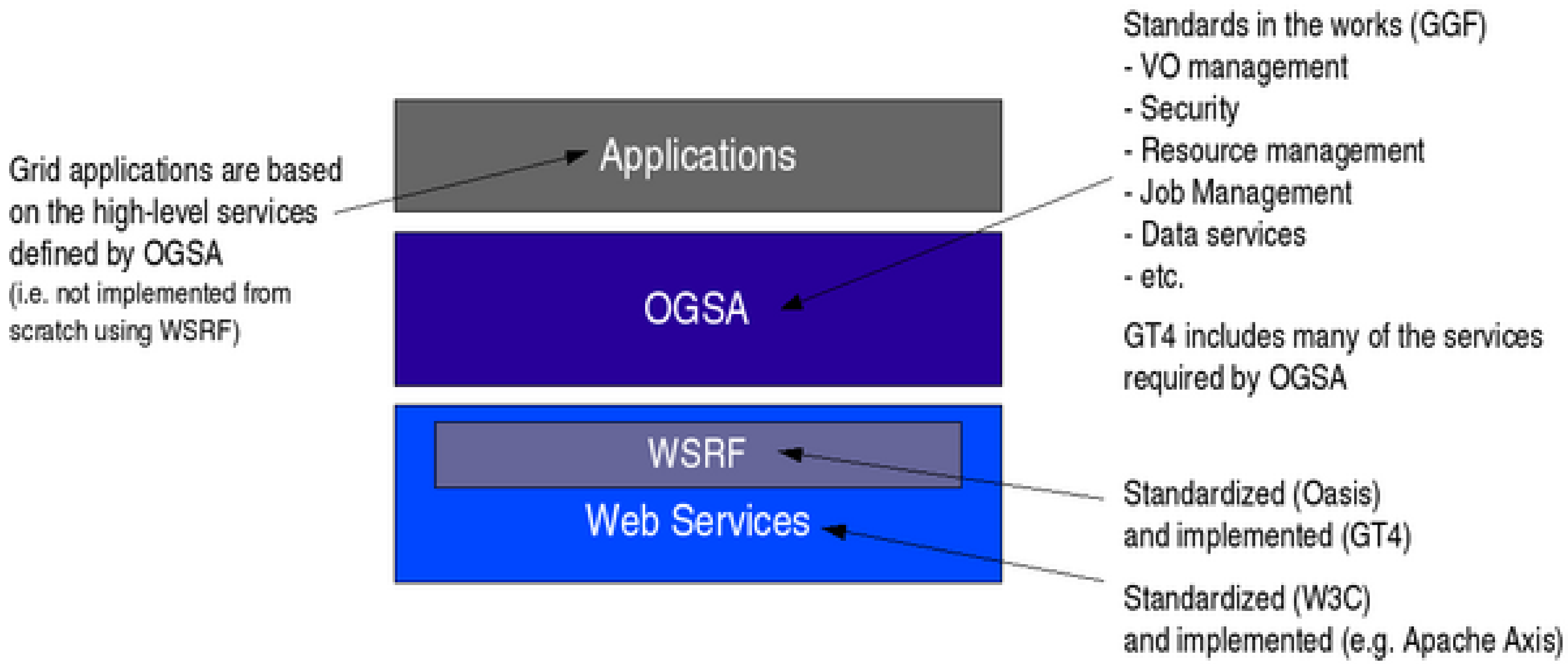
- Too much stuff in one specification
- Does not work well with existing Web services tooling
- Too object oriented

# Grid and Web Services: Convergence

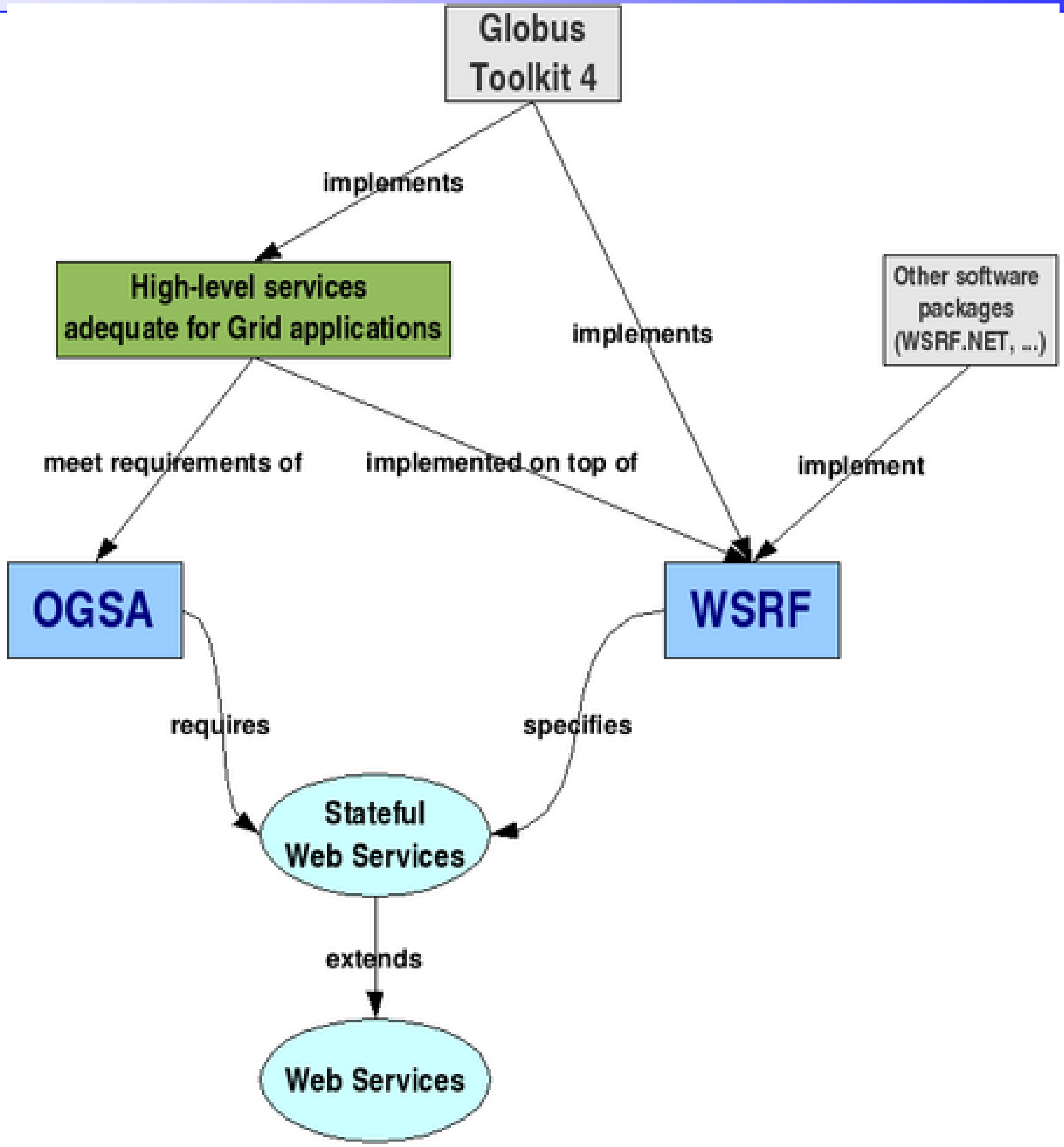



The definition of WSRF means that Grid and Web communities can move forward on a common base

# Layered diagram of OGSA, GT4, WSRF, and Web Services



# Relationship between OGSA, GT4, WSRF, and Web Services



# Towards GT4 production Grids

**Core members:**

- Manchester
- CCLRC RAL

} Data clusters

- Oxford
- Leeds

} Compute clusters

- CSAR
- HPCx

} National HPC services

**Partner sites**

- Bristol
- Cardiff
- Lancaster
- **UoW (Univ of Westminster)**

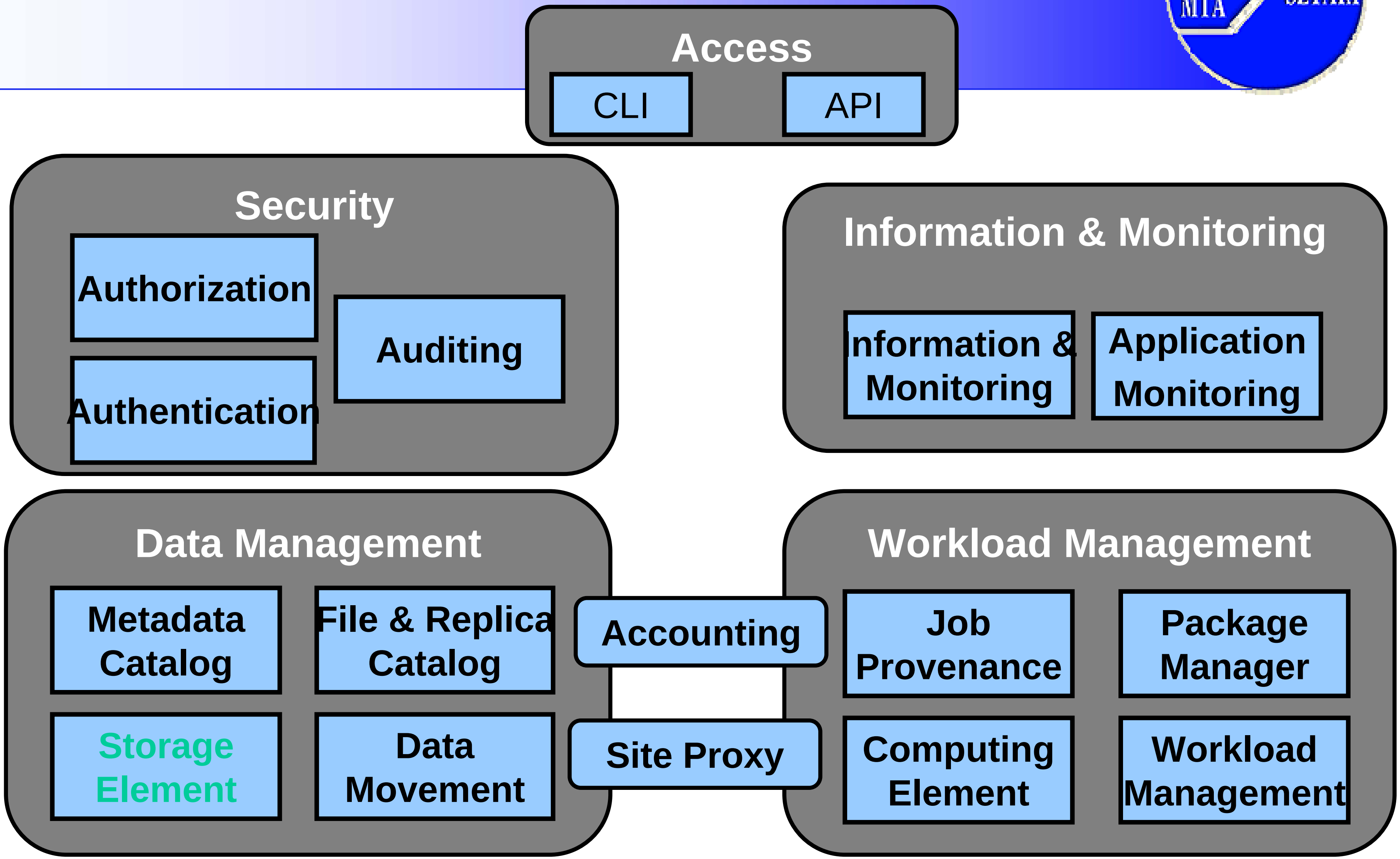
Stable highly-available GT2 production Grid

**Extension with GT4 site and services by UoW**

NGS National Grid Service

core production computational and data grid

# gLite Grid Middleware Services




**Overview paper** http://doc.cern.ch//archive/electronic/egee/tr/egee-tr-2006-001.pdf

# Conclusions

- Fast evolution of Grid systems and middleware:
  - **GT1, GT2, OGSA, OGSI, GT3, WSRF, GT4, …**
- **Current production scientific Grid** systems are built based on 1st and 2nd gen. Grid technologies
- **Enterprise Grid** systems are emerging based on the new OGSA and WSRF concepts