# Grid Computing

**Win Bausch**

**Information and Communication Systems Research Group**

**Institute of Information Systems**

**ETH Zurich**

# Outline

- **The concept of „Grid Computing“**
  - Definition
  - Application domains
  - Taxonomy and Basic Architecture
- **Existing Grid Designs & Implementations**
  - Today‘s Web-based Supercomputers
  - The Globus toolkit: Essential Grid Services
  - WebFlow: Visual Grid Programming using Globus
  - Legion: Object Orientation and Grids
  - Computational Economy
- **Conclusion**
  - Related work at IKS
  - Summary and Outlook

# Defining Grid Computing

- **„The use of (powerful) computing resources transparently available to the user via a networked environment“ [Catlett, Smarr, '92]**
- **The term suggests that using computing resources all over the world will become as natural and pervasive as using the electrical power grid.**
- **Synonyms: seamless, scalable or global computing.**

# Grid Systems Taxonomy

Grand Challenge Problems
Batch Job Processing

→ Computational Grid
- High Throughput Comp.
- Dist. Supercomputing

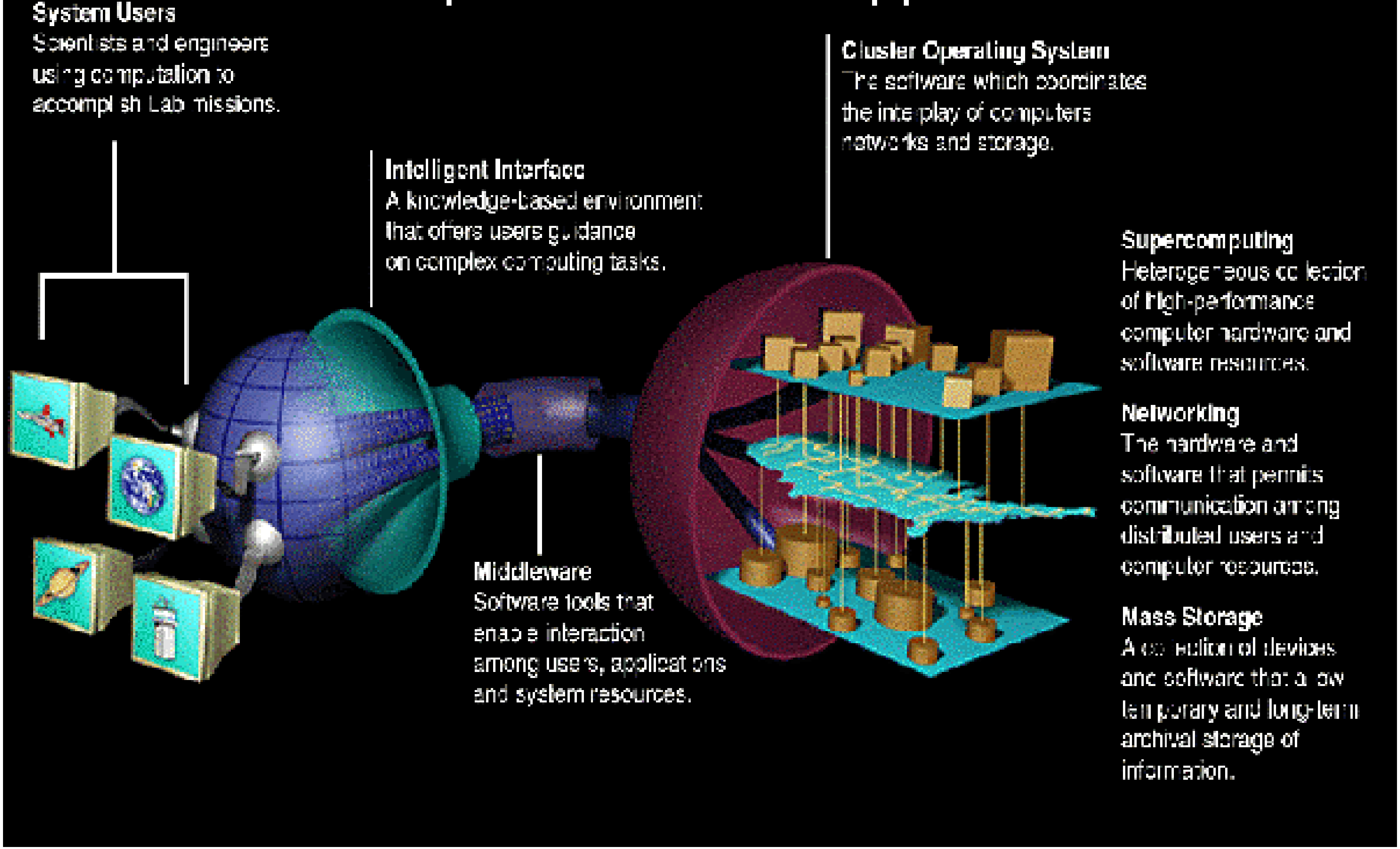
Data Exploration

→ Data Grid

Collaborative Eng./Res.
Multimedia Apps
Highly Adaptive Apps

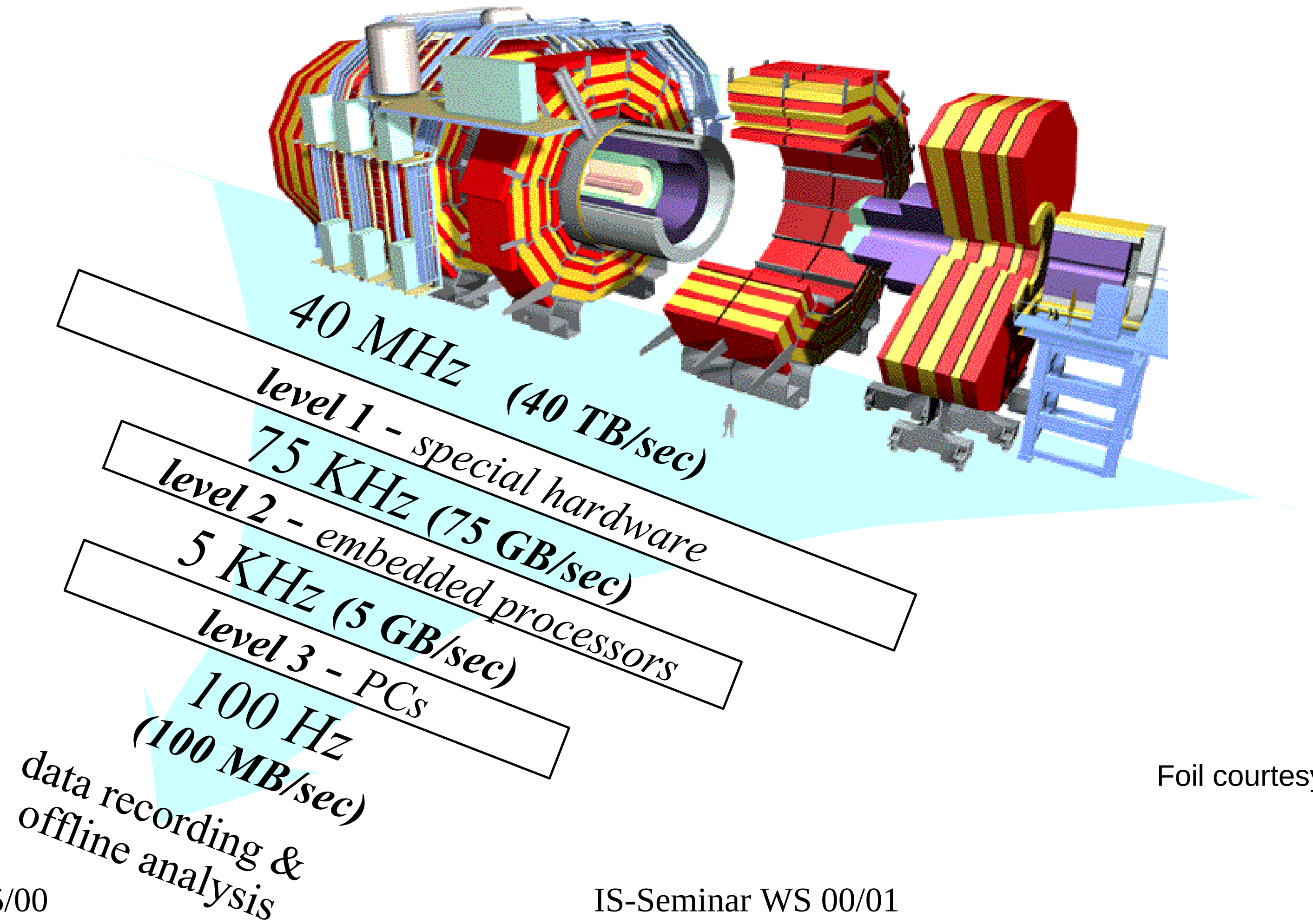
→ Service Grid
- on demand
- collaborative
- multimedia

# Computational Grid Application



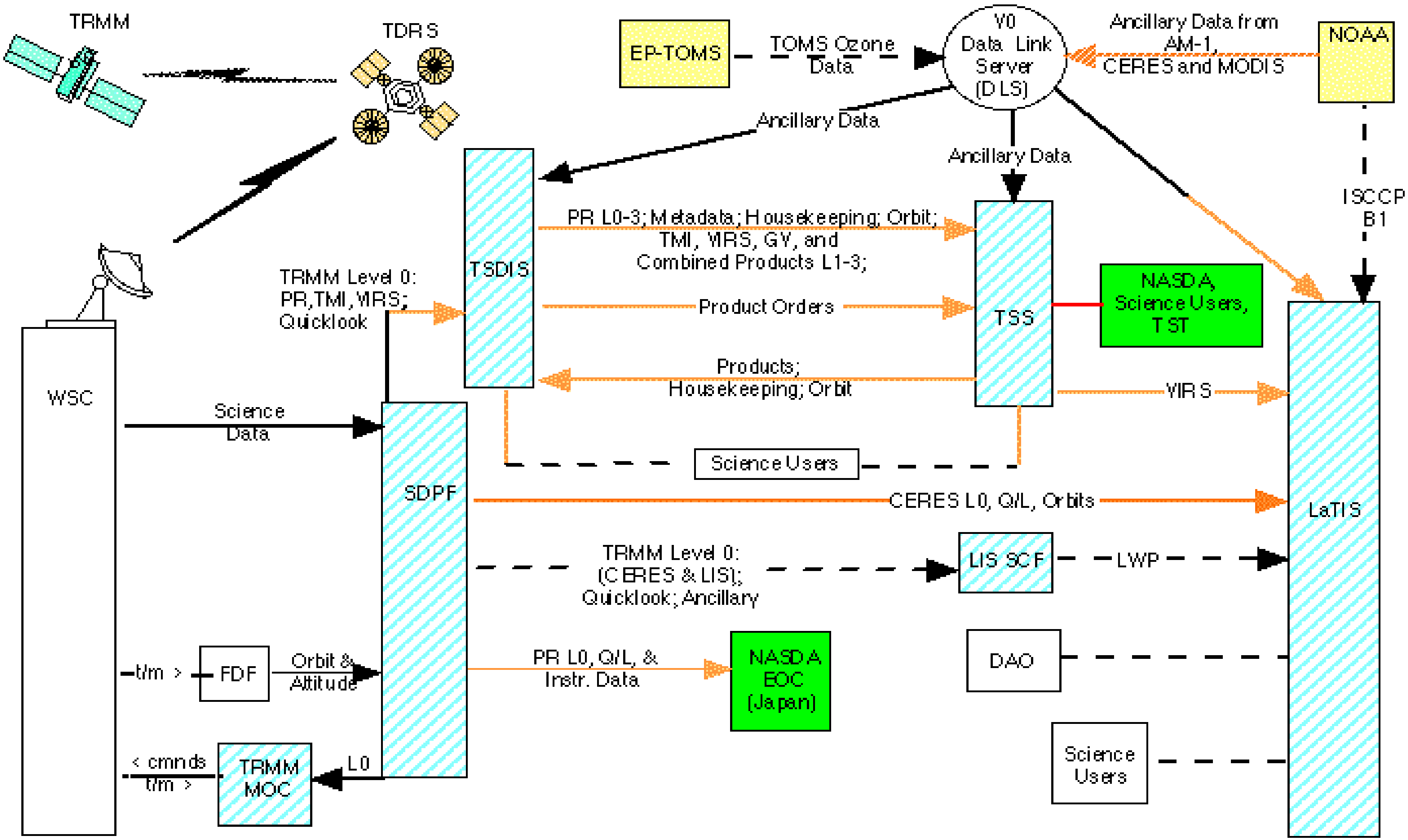
# Data Grid Application

CMS
Compact Muon Solenoid




Foil courtesy of Javier Jaen, CERN

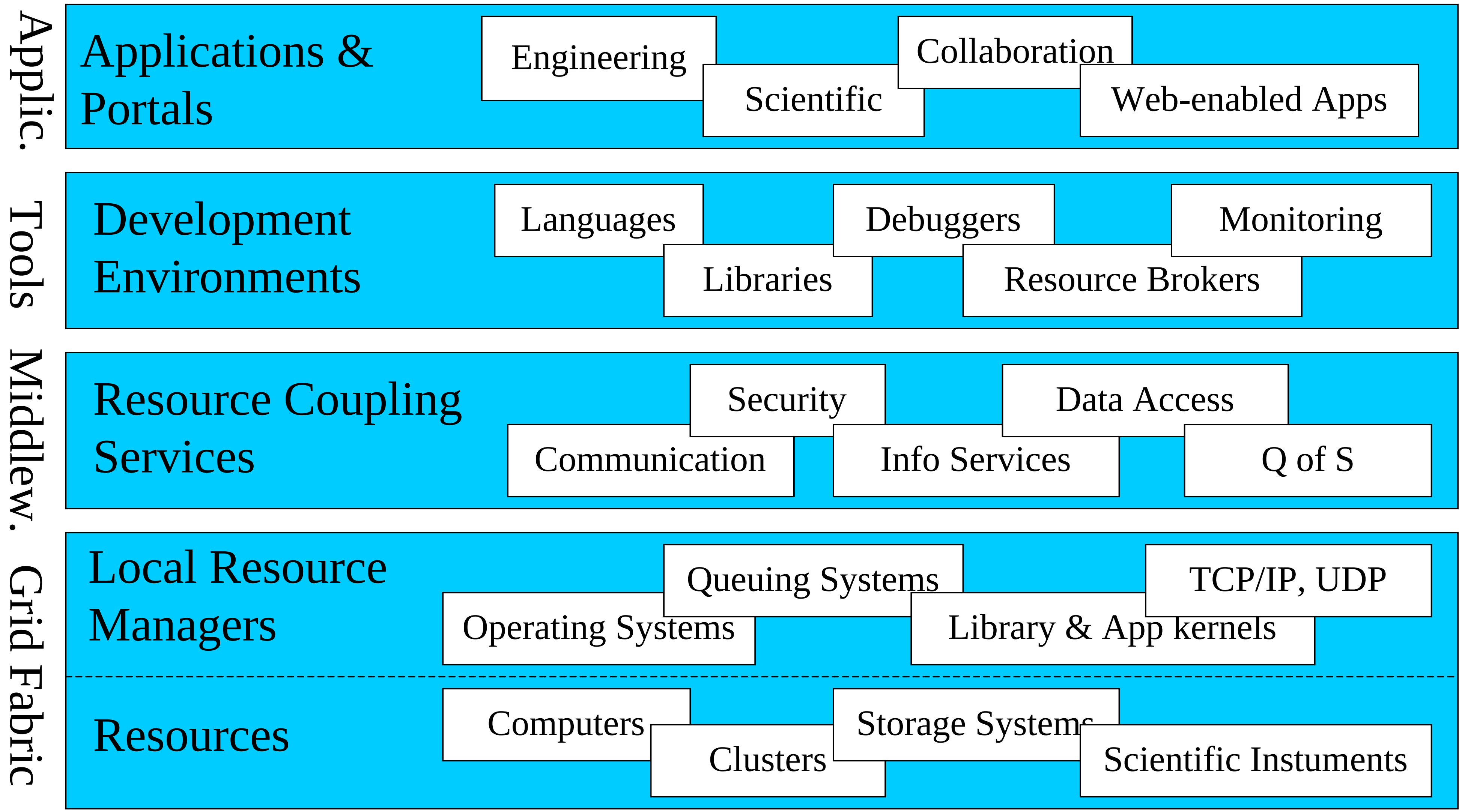
# Service Grid Application



# Grid Key Characteristics

- **Scalability**
  - Since we want to take advantage of the growing network infrastructure.
- **Adaptability**
  - Failure is the rule, not the exception.
  - We do not want to interfere with existing site autonomy.
- **Component Interoperability**
  - Operating systems
  - Communication Protocols
- **All-purpose virtual computer**
  - Avoid mandatory programming paradigm.
  - Grid components have to be flexible/extensible. (RMS, Communication protocols,...)

# Grid Architecture



# RM Design Issues

**Resources**

- **Resource Organization**
  – Flat, Hierarchical, Cell-based
- **Namespace**
  – Relational, hierarchical, graph
- **Resource model**
  – Schema / Object model
- **Resource Info Store Organization**
  – Network directory, Dist. Object Model
- **Resource info dissemination**
  – Periodic (Push/Pull), On demand
- **Resource discovery**
  – Queries (dist./centr.), agents
- **QoS support**
  – None, soft, hard

**Scheduling**

- **Scheduler organization**
  – Centralized, Hierarchical, Decentralized
- **State estimation**
  – Predictive, Non-predictive
- **Rescheduling**
  – Periodic / Event-Driven
- **Scheduling policy**
  – Fixed, Extensible

# Today´s Web-Based Supercomputers

- **Look at The Web as massively parallel machine that is idle most of the time**
- **Market for CPU cycles seems to be emerging**
  - Cost reduction (compared to supercomputers)
    - 1 CPU-year (PII, 400Mhz) will cost around 1500 USD
    - Supercomputer cycles cost around 5 times as much
  - The Web is less capital intensive
  - The Web is permanently renewing itself
- **How does it work?**
  - A company acts as broker between cycle bidders and buyers
  - This company is providing the framework to run the cycle buyer's computation in parallel and takes care of accounting for used CPU cycles on behalf of the cycle bidder.

# Examples

- **Seti@home (setiathome.ssl.berkeley.edu)**
  - Analyze radiotelescope data
- **distributed.net (www.distributed.net)**
  - Break encryption schemes (RSA)
- **Popular Power (www.popularpower.com)**
- **ProcessTree Network (www.processtree.com)**
- **Parabon Computing (www.parabon.com)**

# Important Open Questions

- **Security**
  - How to protect the computation from being maliciously altered?
  - How to deal with security barriers (e.g. firewalls)?
- **Programming the virtual supercomputer**
  - Today‘s candidate applications are mostly embarassingly parallel computations. What about more complex computations?
- **Business model**
  - Does CPU cycle brokerage economically make sense? (too many bidders, not enough buyers)
  - Upcoming Computational Grid Systems may render „cycle brokers“ - which are mediators - obsolete.

# The Globus Toolkit

- **Low-level toolbox for building a grid. Provides modules for:**
  - Resource allocation, process management
  - Resource reservation
  - Uni- and multicast communication services
  - Authentication & security
  - Grid information services (structure/state)
  - Health monitoring of system components
  - Remote data access (sequential or parallel)
  - Executable construction, caching and location
- **Emphasis is on providing generic, orthogonal services that can be used to implement higher-level services, which in turn are used by grid application software.**

# The Globus RM Design

- **Machine organization**
  - Hierarchical Cell
- **Resource Model**
  - Schema model
  - Hierarchical namespace
  - Network Directory Stores
  - Soft QoS
  - Distributed Query Resource Discovery
  - Periodic Push Resource Information Dissemination
- **Scheduling**
  - Low-level services like reservation, co-scheduling

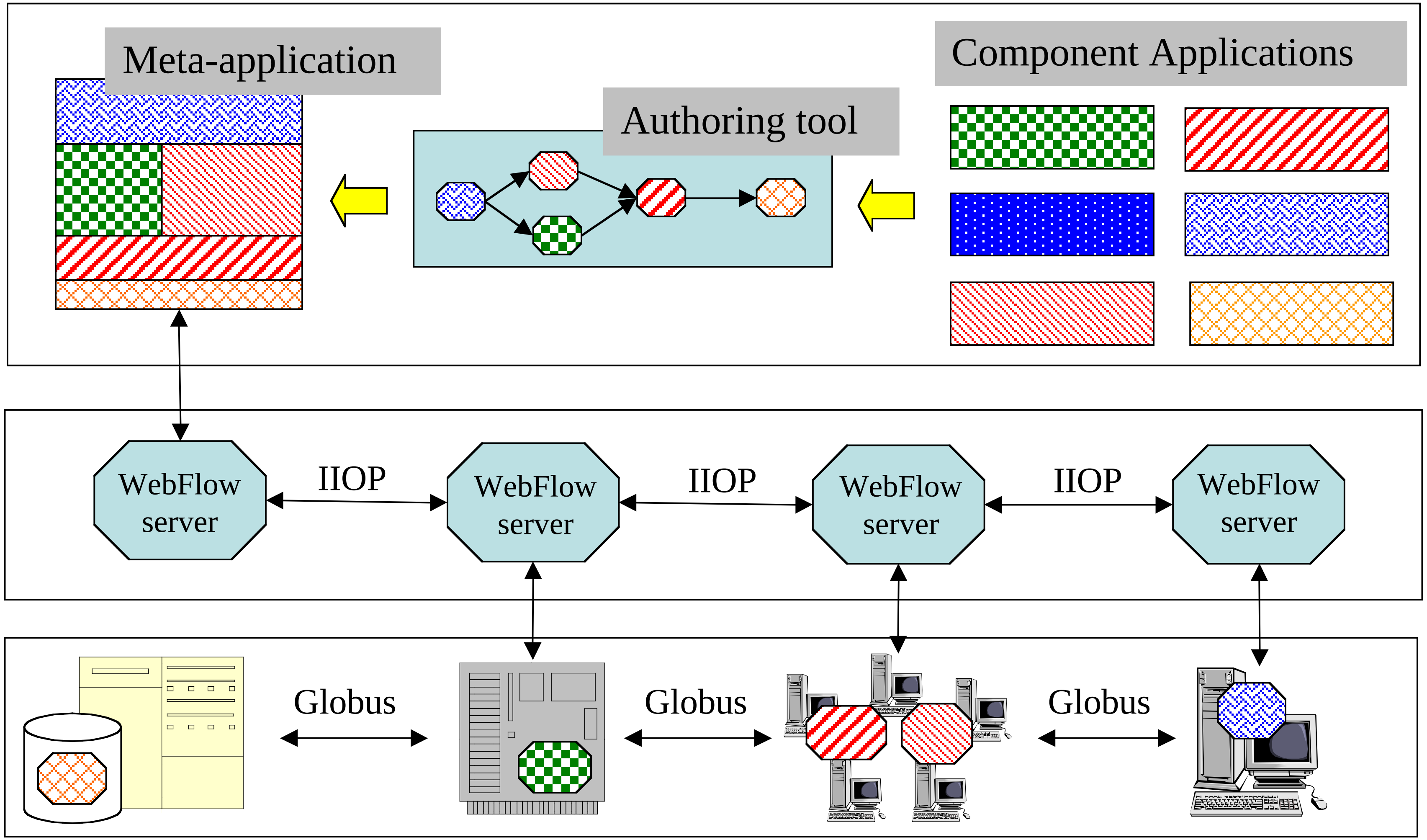
# Simple Globus Grid

- **Sign-on / Sign off**
- **Run programs on remote hosts**
  - Rsh-like, executable location has to be specified additionally
  - Submission to Batch Processing System (PBS)
  - MPI programs, degree of parallelism provided on command line
  - Job scripts can be written using Globus RSL
- **Add/Remove/Query sites**
  - Simple data filters for querying
- **Move data between sites**
  - Globus Remote Copy, works using a Globus data server running on the source and destination nodes
  - Copying via http(s) also supported

# Visual Programming with WebFlow

- **Extend the web model so as to allow wide area distributed computing**
- **3 key ideas:**
  - „Publish“ reusable computational modules on the Web.
    ( modules analogous to web pages)
  - Programming the grid consists in connecting different modules using data flow connectors.
    (data flow links analogous to hyperlinks)
  - Use visual authoring tool to do this.
- **Implementation**
  - Middle tier is java servlet-based (Apache web servers).
  - CORBA provides fault tolerance in the middle tier.
  - Backend tier based on Globus toolkit.

# WebFlow Architecture



# Legion: Object Orientation in The Grid

- **The advantage of Legion is that every grid element is represented by an object:**
  - Solves the interoperability problem.
  - Reduces system complexity.
  - Fault containment is easier to achieve.
  - Inheritance enables software reuse.
  - Access control can be done at object boundaries. Resource owners decide on access policy when designing/implementing the objects.
- **The disadvantage of Legion is that every grid element is represented as an object:**
  - It is difficult to wrap legacy code. (What is the best object-oriented model for the shared memory paradigm?)
  - Every grid element has to be wrapped. This is a non-negligible amount of work since legacy code typically has procedural interfaces.

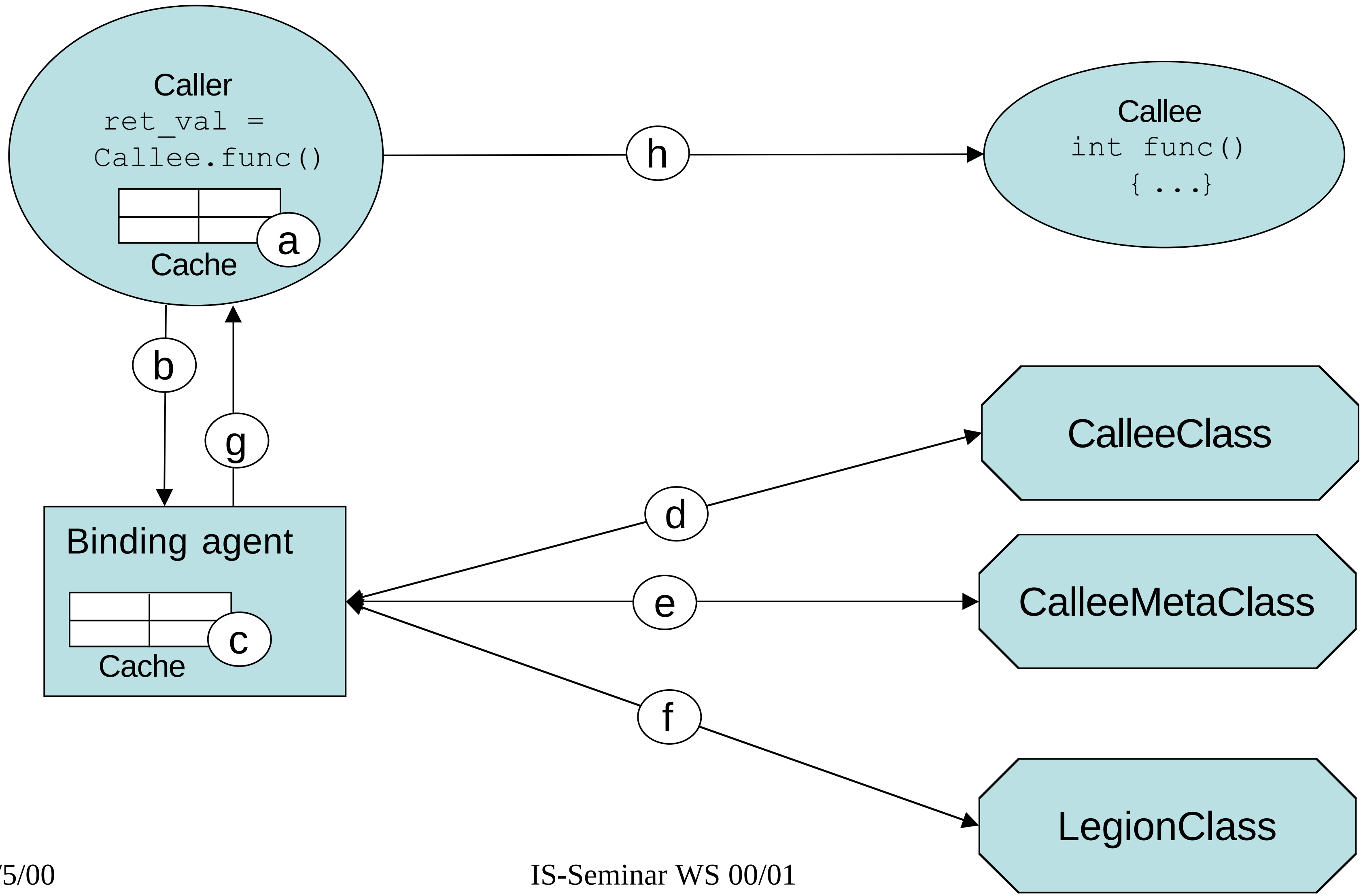
# Legion RM Design

- **Machine organization**
  - Any
- **Resource Model**
  - Object Model
  - Graph Namespace
  - Object Model Store
  - Soft QoS
  - Distributed Query Resource Discovery
  - Periodic Pull Resource Information Dissemination
- **Scheduling**
  - Hierarchical scheduler, ad-hoc extensible scheduling policies

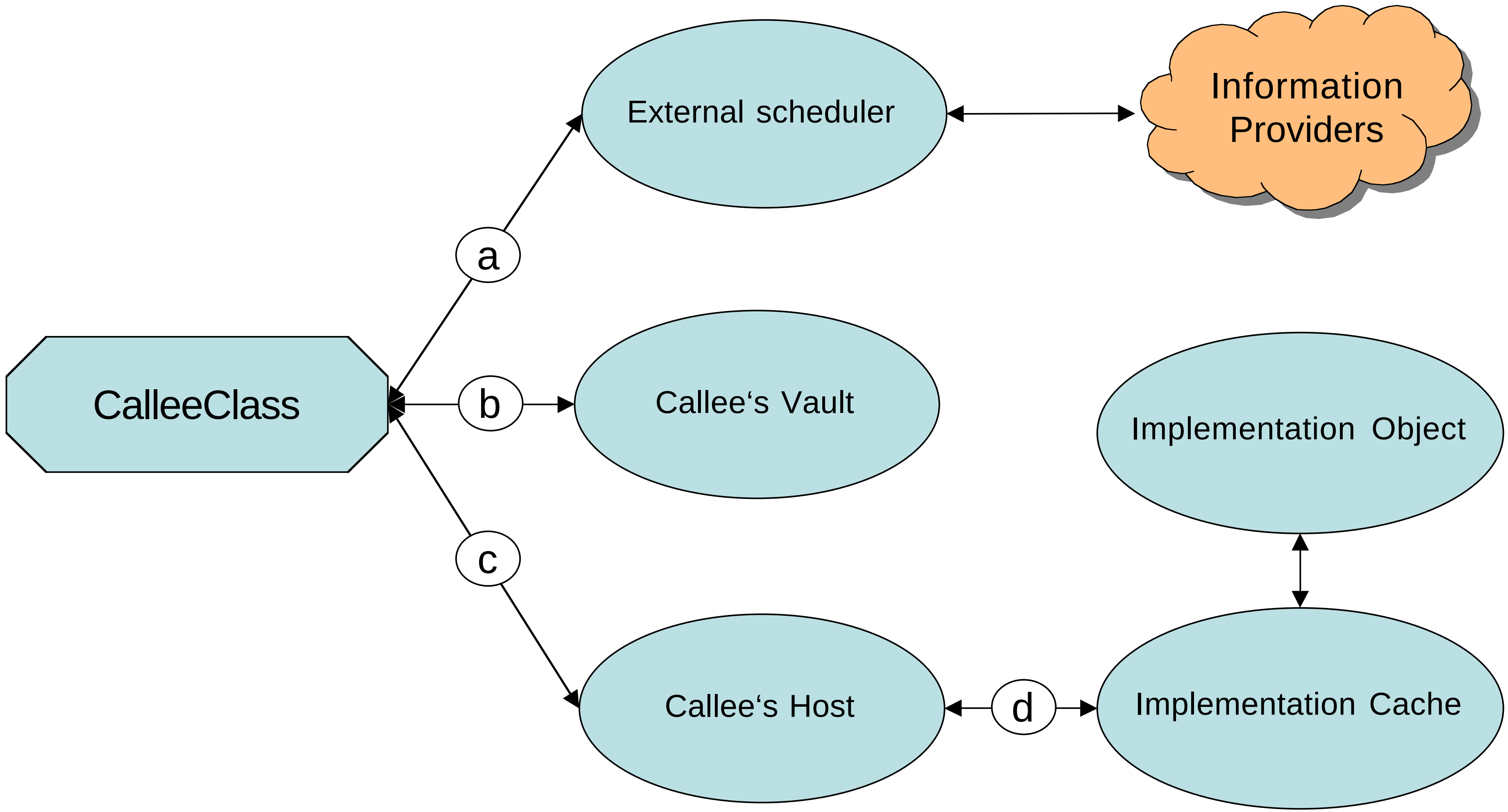
# Legion Architecture

- **Grid elements are one of the following:**
  - Host objects: encapsulate computing resources
  - File objects: encapsulate storage space
  - Implementation objects : encapsulate executables
  - Implementation caches : encapsulate collections of executables
  - Vault objects: encapsulate persistent storage for stateful objects
  - Binding agents : encapsulate namespace implementation
  - User-defined classes : encapsulate steps of the computation
- **Legion classes play two special roles:**
  - They manage their instances (location, activation, deactivation) and know their derived classes.
  - They act as policy makers (when to activate/deactivate objects)
- **The namespace constists of**
  - Context names (to make life easier for grid programmers)
  - LOID‘s, which are unique identifiers an, among other characteristics , encode the inheritance hierarchy of the object they identify.
  - OA‘s which are „physical adresses“
  - The translation of LOID‘s to OA‘s is done by binding agents.

# LOID Binding



# Instance Activation

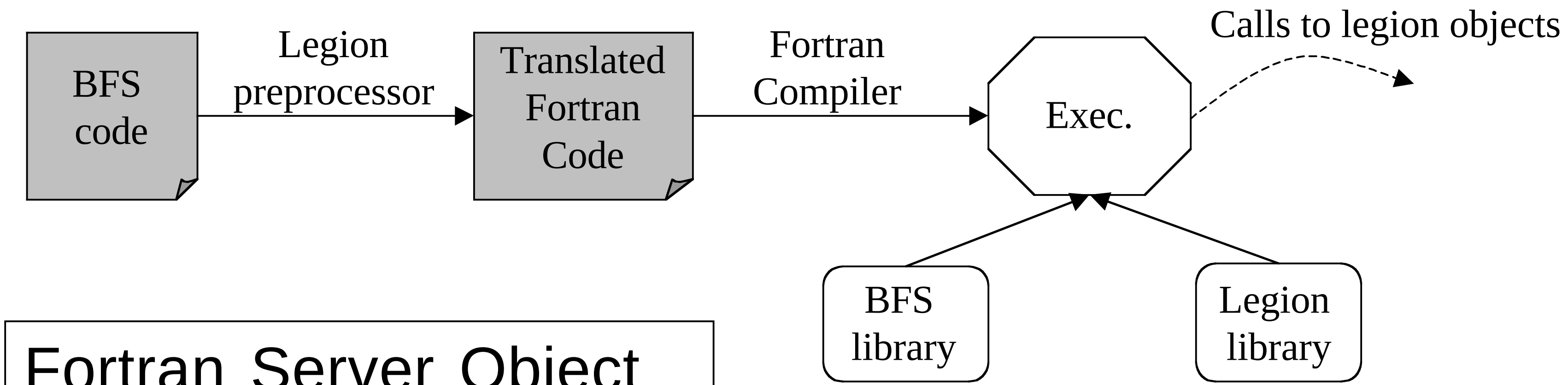


# Building a Legion (I)

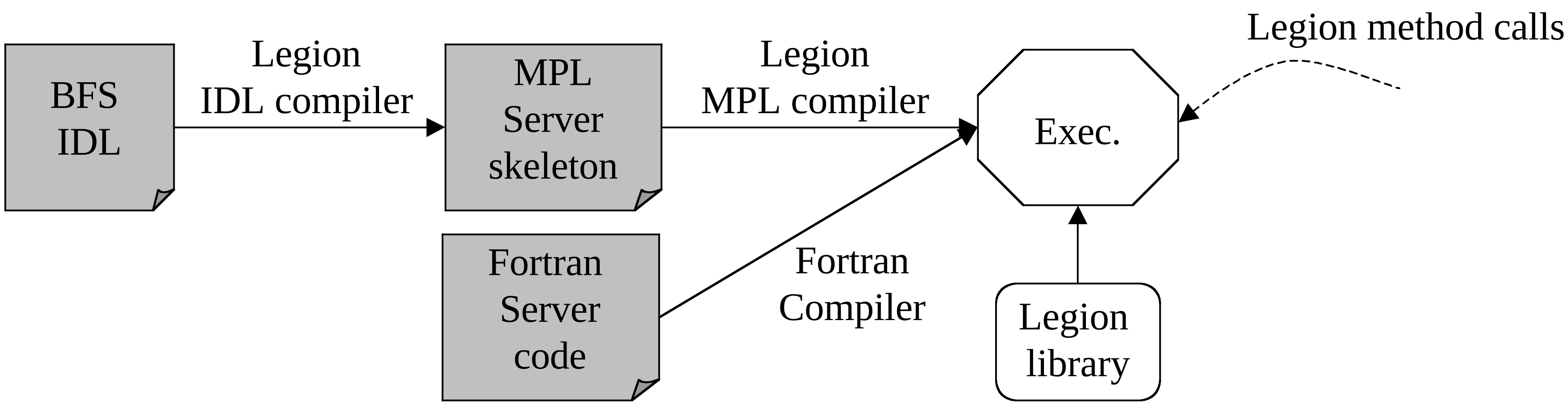
- **Legion objects are implemented using the Mentat Programming Language.**
- **MPL...**
  - ...is an extension of C++.
  - ...was developed to hide low-level parallelism from the programmer. The compiler takes care of synchronizing parallel code by detecting control and data flow dependencies in parallel code and adds appropriate code for synchronization and communication.
- **Special support is provided for Fortran, simplifying the development of Fortran implementation objects for Legion.**

# Building a Legion (II)

## Fortran Client Application



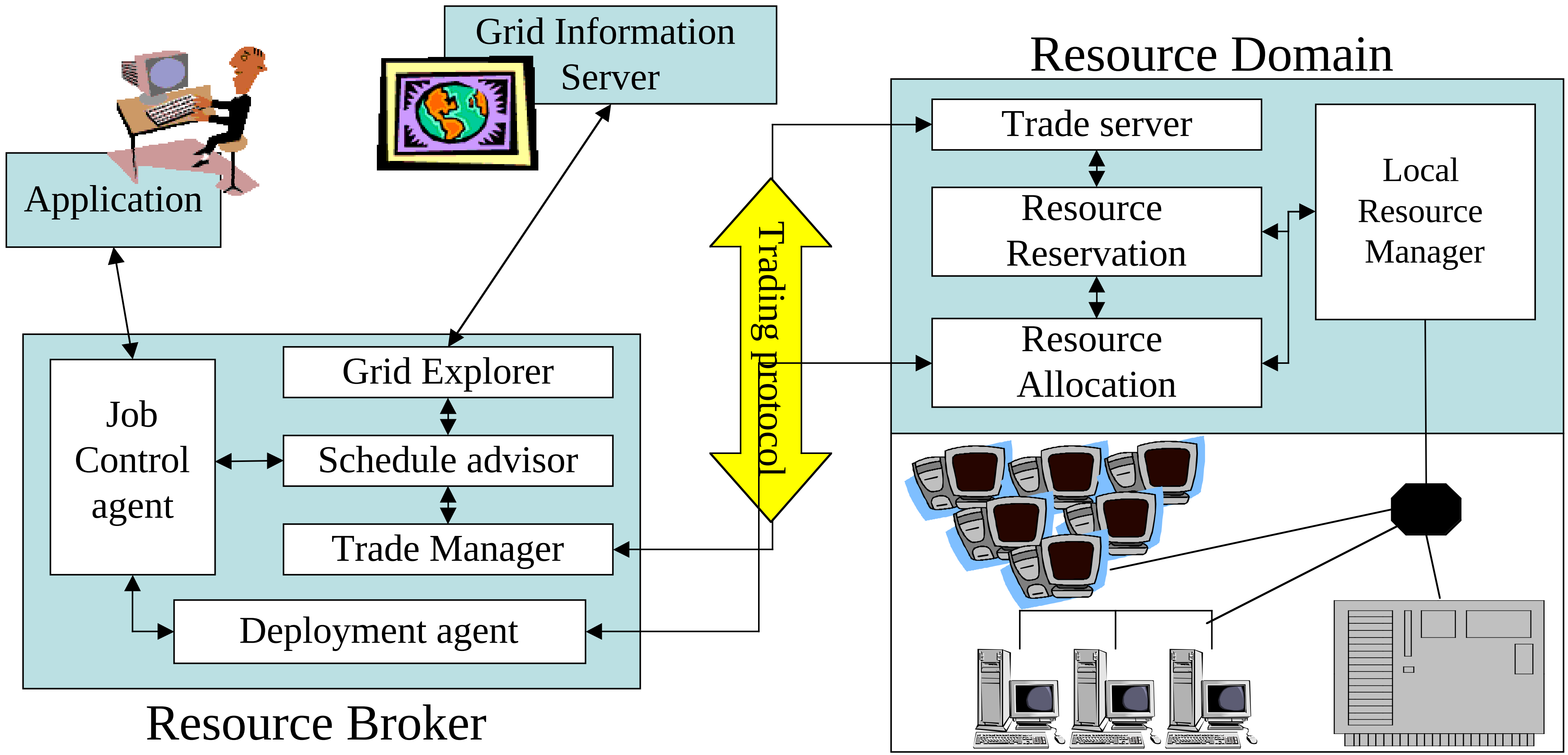

## Fortran Server Object



# Computational Economy

- **So far, the grid has not been used commercially, implying that contributors did not expect a profit, except fame, or maybe fun...**
- **This will change in the future, requiring other incentives to be created.**
- **Proponents of computational economy believe this incentive can and will be created by making the grid support resource trading.**
- **In a computational economy, grid users want to minimize the cost of their computation, whereas resource owners want to maximize their profit.**

# GRid Architecture for Computational Economy



# BioOpera: Structuring coarse grained computations

- **The IKS Research Group develops the BioOpera prototype system to help bioinformaticians to...**
  - ...rapidly integrate existing tools into bigger applications
  - ...simplify deployment and migration of these applications (compared to scripting languages commonly used as „glue“)
  - ...dependably run the computations on a COW
- **Computations are specified in terms of control flow dependency graphs (similar to WebFlow)**
- **We have been sucessfully running a month-long computation on a cluster of COTS with minimal human intervention and in spite of node failures.**

# Conclusion

- **There are a wide number of different approaches to grid computing.**
- **It is not known which approches are best. This statement is backed up by the fact that the „Grid Forum“ is pushing towards developing and documenting „best practices“.**
- **A lot of efforts are made in the middle and backend tiers to provide interoperability, fault-tolerance, etc.**
- **However new concepts will also have to be developed to allow end-users to exploit the variety of resources that future grids will offer („Human-Grid interface“).**
  – How does a user detect resources with certain capabilities?
  – What is the best way to structure and represent grid applications?
  – How do we represent the notion of „cost“ of a complex distributed computation?