Grid Computing:

Status and Perspectives

Alexander Reinefeld
Florian Schintke

„Schwerpunkte der Informatik"
Ringvorlesung am 05.06.2003

Outline

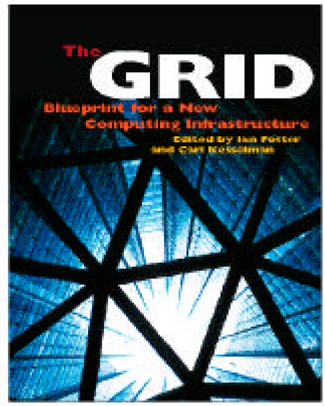
- **MOTIVATION**
  - What‘s a Grid? Why using Grids?
- **TWO TYPICAL APPLICATION DOMAINS**
  - Computation Fluid Dynamics: FlowGrid
  - Particle Physics: DataGrid
- **WHAT'S MISSING IN CURRENT GRID SYSTEMS**
  - Fault tolerance
  - Scalability
  - Distributed data management
  - Advance reservation, co-scheduling
- **CONCLUSION**

What's a Grid?

"A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities."

Ian Foster, Carl Kesselman (1999)

strongly influenced by early metacomputing projects!



What's a Grid? (2nd Attempt)

Grid computing is concerned with *"coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations."*

I. Foster, C. Kesselman, S. Tuecke, „Anatomy of the Grid", 2000

-> Stresses importance of **protocols** as a means of enabling **interoperability**

now also addresses social and policy issues

What‘s a Grid? (3rd Attempt)

Ian Foster‘s Three-Point Checklist in hpcwire (22.07.2002):

„A Grid is a system that

- *coordinates resources ... at are not subject to centralized control*
- *uses standard, open, ge... purpose protocols and interfaces*
- *delivers nontrivial qualitie... e“*

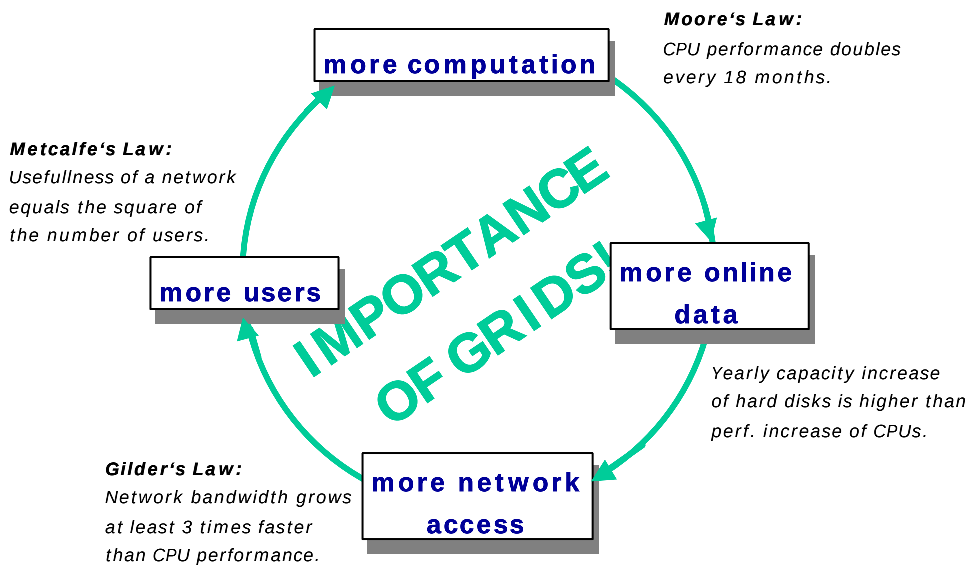
„Ian's checklist may be more **valid for some very large research Grids**, in the future. But this checklist would exclude many contributions from industry, thus likely pushing Grid computing into a niche.“

W. Gentzsch, hpcwire 05.08.2002

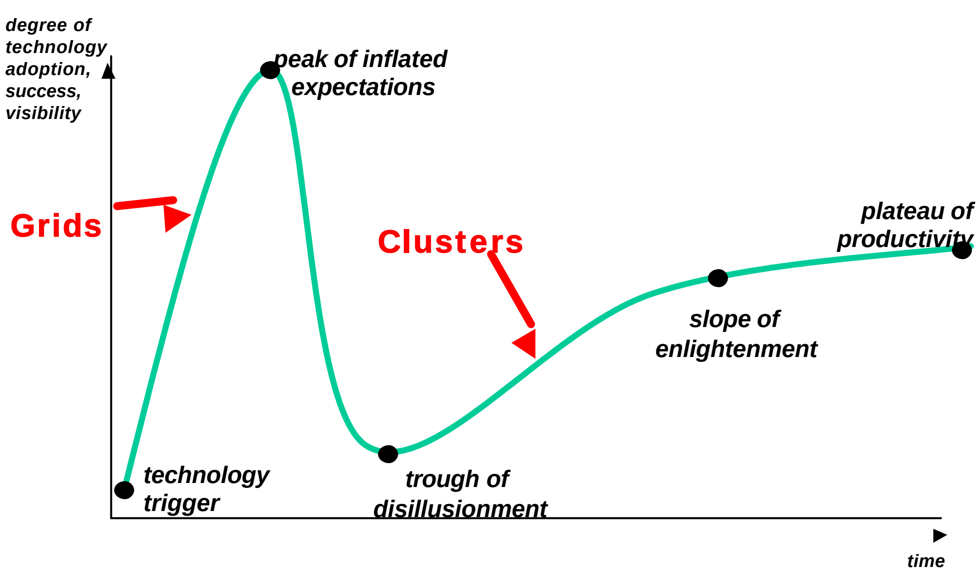
What‘s a Grid? - Our Definition

- Grids are **large**
  - in terms of potentially available resources and distance between them
- Grids are **distributed**
  - substantial latencies in moving data, may dominate the application
- Grids are **dynamic**
  - resources may change during the lifespan of an application
- Grids are **heterogeneous**
  - form and properties of sites (nodes) may differ significantly
- Grids are **across boundaries of organizations**
  - access policies differ at different sites

Why Using Grids?



Emerging Technology Hype Curve

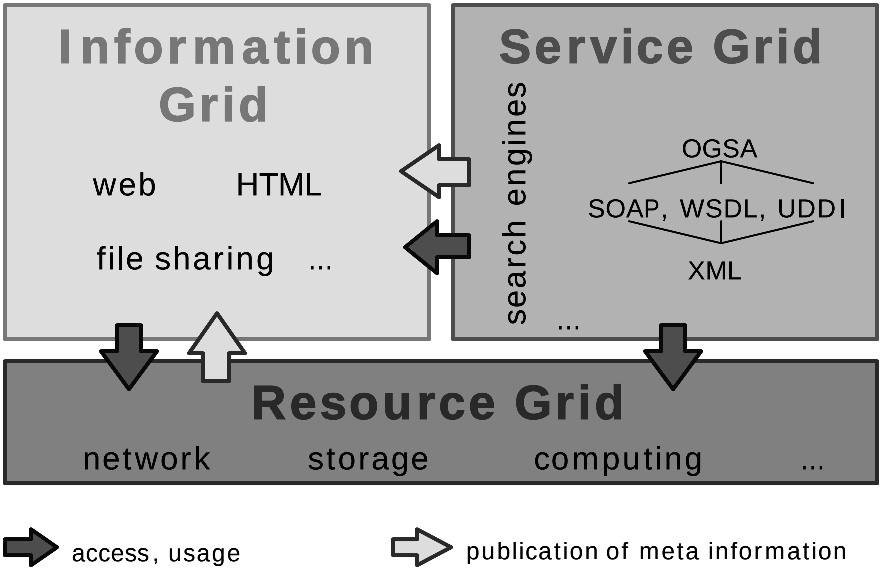


3 Approaches to Building Grid Environments

- **a collection of tools bundled in a „toolkit“ (like Globus 2.0)**
  - provides access and protocols to use distributed computing resources efficient
  - applications are loosely coupled
  - **emphasis on openness**
- **a distributed operating system (like Legion)**
  - supports OO distributed programming
  - applications are tightly coupled
  - **emphasis on coherence**
- **a distributed resource management framework (like Condor)**
  - harvesting resources for high-throughput computing
  - **emphasis on work-load balancing**

Many Projects - What did we learn?

What‘s a Grid? Our View

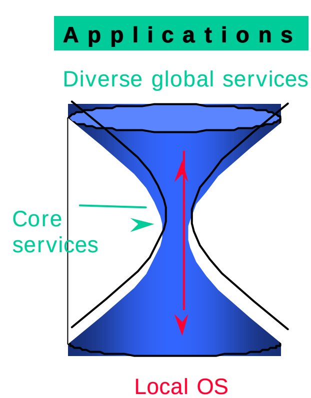


Example of a <u>Resource Grid</u>: Globus 2.0 Toolkit™

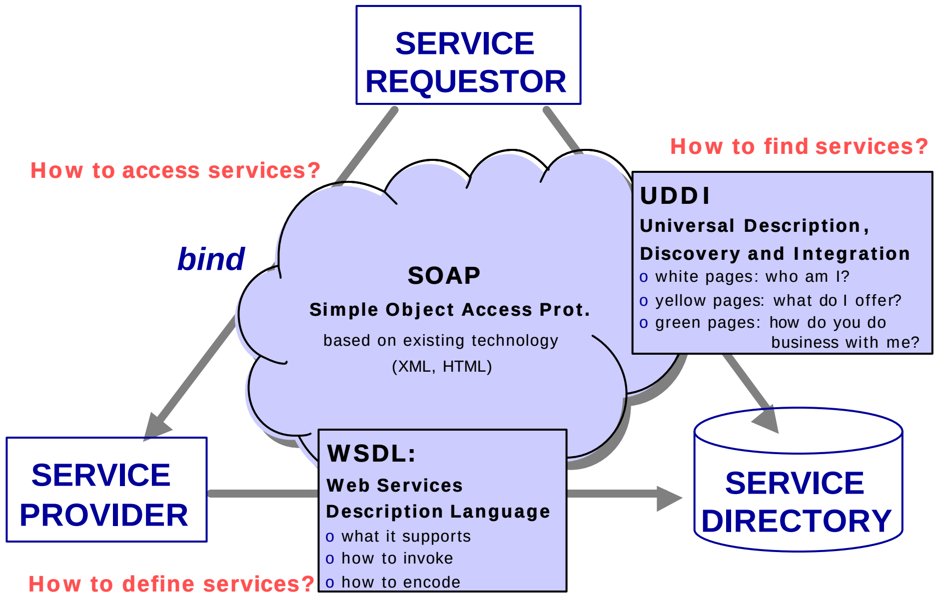
GLOBUS' PROTOCOLS

- **Connectivity layer**
  - o **GSI:** Grid Security Infrastructure
- **Resource layer**
  - o **GRIP:** Grid Resource Information Protocol
  - o **GRAM:** Grid Resource Allocation Management
  - o **GridFTP:** Grid File Transfer Protocol
- **Collective layer protocols**
  - o Info Services, Replica Management, etc.

Hourglass Model



Towards a Service Grid: Web Services



WEB Services versus GRID Services

- **WEB Services** address discovery & invocation of **persistent services**
  - Interface to persistent state of entire enterprise
- **GRIDs** must also support **transient service instances** that are dynamically created/destroyed
  - need interfaces to the states of distributed activities

Significant implications for how services are managed, named, discovered, and used

OGSA
Open Grid Service Architecture

OGSA - Framework for Grid Services

- Each OGSA-compliant service should support standard interfaces for:
  - **creation of service instances** (Factory)
  - **lifetime management** (Get-/SetTerminationTime, explicit destruction)
  - **registration & discovery** (Query, extensible query language)
  - **authorization** (remote access control policy management)
  - **notification** (observe service existence, lifetime, information, ...)

SOAP, WSDL, and UDDI ***can*** build the base for an OGSA implementation.

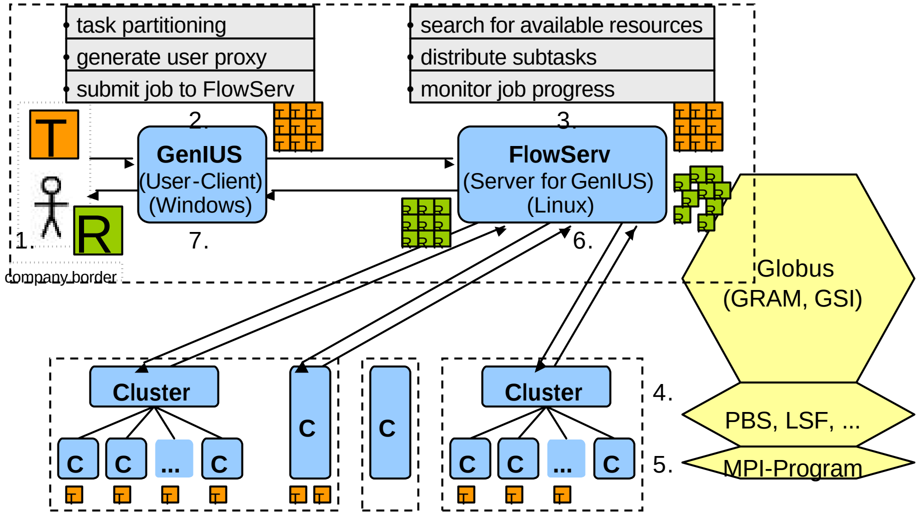
Globus3.0 (GT3) contains OGSA compliant services for all (most?) GT2 services

Outline

- MOTIVATION
  - What's a Grid? Why using Grids?
- **TWO TYPICAL APPLICATION DOMAINS**
  - Computation Fluid Dynamics: FlowGrid
  - Particle Physics: DataGrid
- WHAT'S MISSING IN CURRENT GRID SYSTEMS
  - Fault tolerance
  - Scalability
  - Distributed data management
  - Advance reservation, co-scheduling
- CONCLUSION

Example 1: FlowGrid

A typical run, starting with the specification of a CFD task T and ending with the presentation of the results R to the user



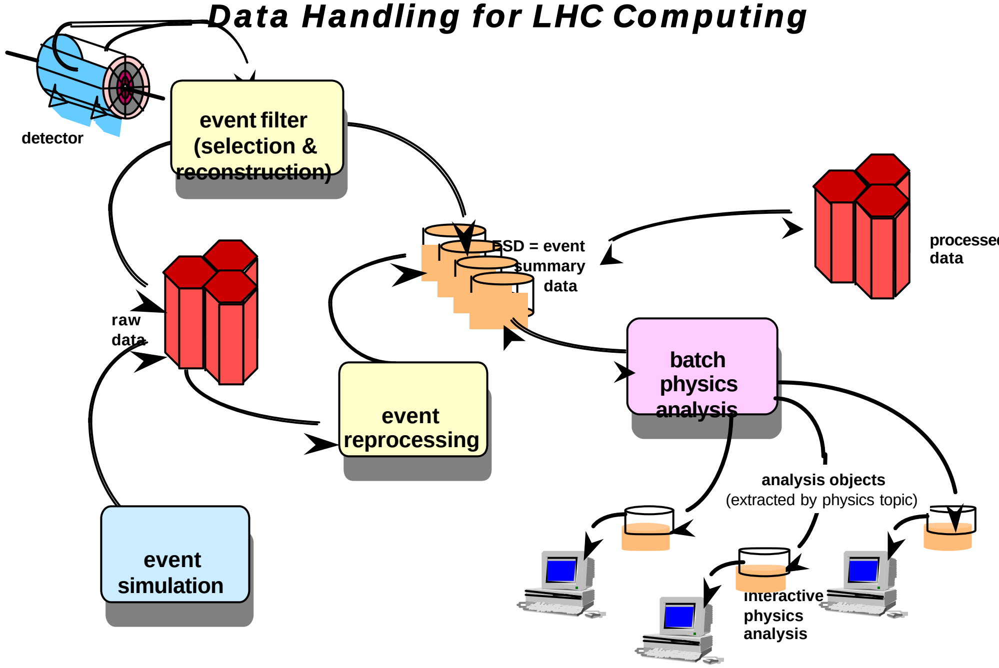
Resource Description in FlowGrid

```
<System SystemSite="Symban">
  <ResourceManager>
    <ManagerName>zeus.flowgrid.com</ManagerName>
    <CPUInfo>
      <CPUSpeed>500</CPUSpeed>
      <CPUCount>2</CPUCount>
      <CPUAvailable>1</CPUAvailable>
    </CPUInfo>
    <MachineMemory >2000</MachineMemory >
    <MachineDiskspace>40000</MachineDiskspace>
    <MachineNetSpeed>100</MachineNetSpeed>
    <InstalledSoftware >genius.exe</InstalledSoftware >
    <InstalledLicense>genius.lic</InstalledLicense>
  </ResourceManager>
  <ResourceManager>
    <ManagerName>hercules.flowgrid.com</ManagerName>
    <CPUInfo>
      <CPUSpeed>2400</CPUSpeed>
      <CPUCount>1</CPUCount>
      <CPUAvailable>1</CPUAvailable>
    </CPUInfo>
    <MachineMemory >1000</MachineMemory >
    <MachineDiskspace>60000</MachineDiskspace>
    <MachineNetSpeed>100</MachineNetSpeed>
    <InstalledSoftware >genius.exe</InstalledSoftware >
    <InstalledLicense>genius.lic</InstalledLicense>
  </ResourceManager>
</System>
```

Example 2: DataGRID

The 4 detectors of the *LHC* at Cern will produce ~10^9 events per year = 4 PB

Data Handling for LHC Computing

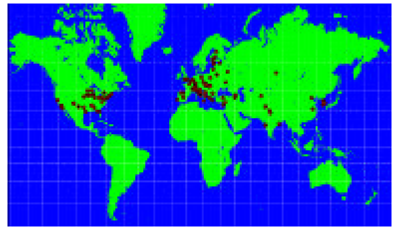


des.robertson@cern.ch

Data Access Pattern

Cern worldwide has

- 267 institutes in Europe, 4603 users
- 208 institutes elsewhere, 1632 users
- all will be accessing the LHC Data Grid!



Layered Architecture

1 Tier-0 center (Cern) with $O(10^4)$ PCs and $O(10^4)$ TByte storage

~ 10 Tier-1 centers with $O(10^3)$ PCs and $O(10^3)$ TByte

> 10 Tier-2 centers with $O(10^2)$ PCs

> 100 Tier-3 centers with some PCs or SMPs

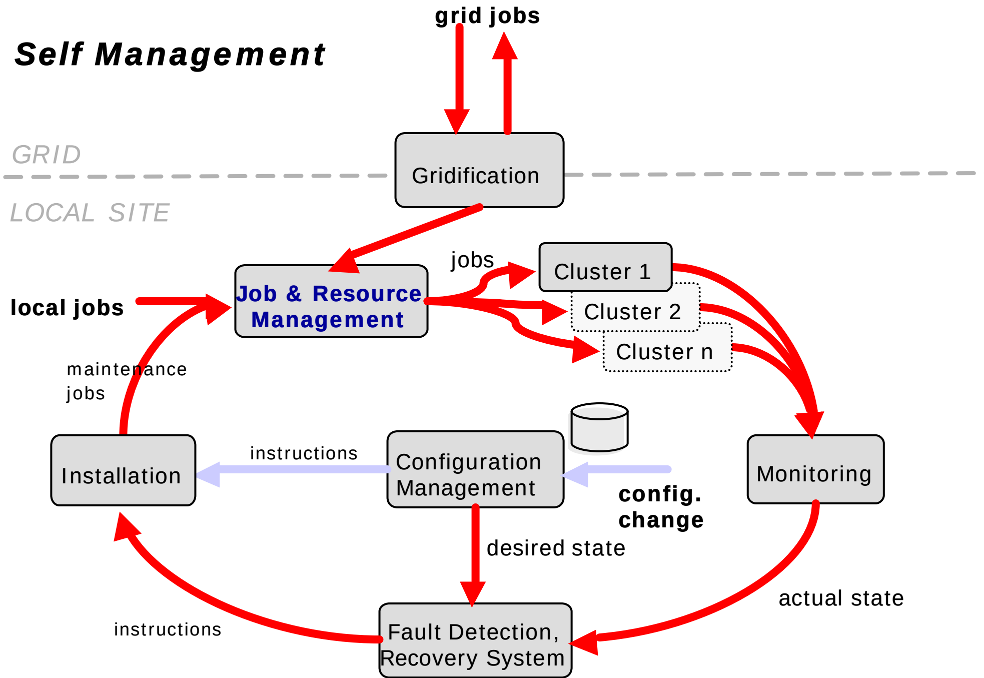
Data stored distributedly

Large PC Farms: Cost Effective? Reliable?

assuming PCs with one 500 GB disk, GbE, 250 Watt, 3000 €

nodes	Tier 1: 3,000	Tier 0: 10,000	all LHC: 50,000
daily faults (avg. PC lifetime is 5 years)	1.6 / d	5.5 / d	27 / d
summed disk storage	1.5 PB	5 PB	25 PB
disk read errors (every 8 y = 70.000 h)	1.0 / d	3.4 / d	17 / d
one bit error per transferred TBit (GbE)	0.2 Mio / d	0.8 Mio / d	4.3 Mio / d
power consumption	750 kWatt	2.5 MWatt	12 Mwatt
power cost (10 cent/kWh)	650 k€/y	2.2 M€/y	11 M€/y
monthly hardware reinvest	144 k€/m	500 k€/m	2.4 M€/m

Self Management



work of Thomas Röblitz, ZIB

Outline

- **MOTIVATION**
  - o What's a Grid? Why using Grids?
- **TWO TYPICAL APPLICATION DOMAINS**
  - o Computation Fluid Dynamics: FlowGrid
  - o Particle Physics: DataGrid
- **WHAT'S MISSING IN CURRENT GRID SYSTEMS**
  - o Fault tolerance
  - o Scalability
  - o Distributed data management
  - o Advance reservation, co-scheduling
- **CONCLUSION**

Scalability

- MDS – the Grid metadata catalog uses hierarchical LDAP servers
  → not scalable for large Grids!

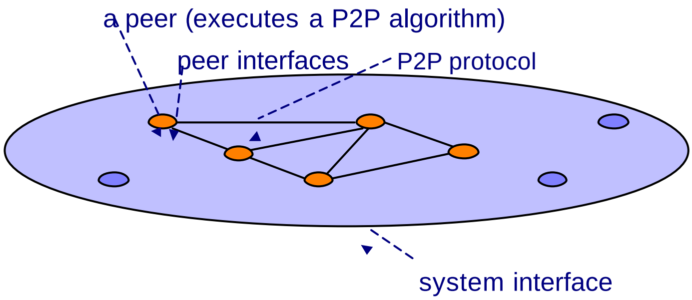
- OGSA based on SOAP calls
  → too slow for HPC environment and massive accesses.

Latency of „remote" DoNothing() call on localhost

System	Language	Latency (ms)
JavaRMI	Java	1.2
CORBA	Java	1.5
MS SOAP Toolkit	Visual Basic	16.8
SoapRMI	Java	19.5
SOAP::Lite	Perl	42.0
Apache SOAP	Java	23.4
Apache Axis	Java	15.6

How to Achieve Scalability?

A P2P System = A Component



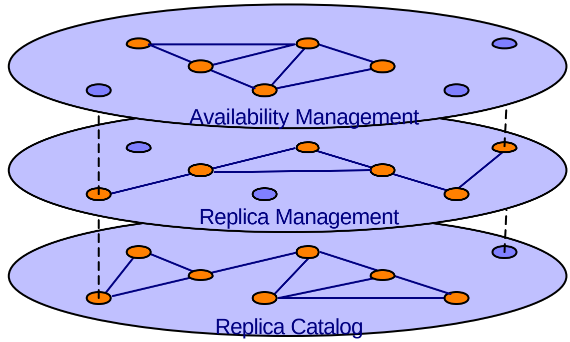
P2P systems are simple.

Together, they may perform complex tasks.

Combining P2P Systems

Components

- have their **own overlay network**
- are **loosely coupled** or **independent**



FUNCTIONAL COMPONENTS provide user functionality

SELF MANAGEMENT COMPONENTS monitor and control the grid

Distributed Data Management

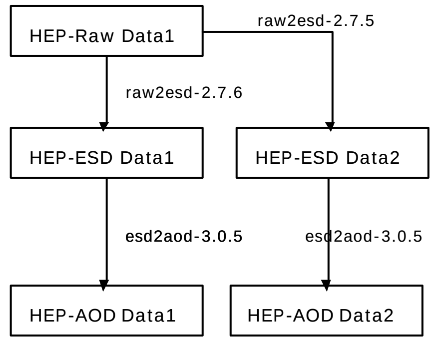
- **replication** to improve reliability by introducing redundant file copies,
- **synchronization** for keeping replicas in a consistent state,
- **placing** to determine optimal replica placement for fast access,
- **caching** and **pre-fetching** to benefit from spatial and temporal locality,
- **staging** to improve job execution time by scheduled data transfers,
- **locating** nearest replica for fast processing.

Metadata Management and Co-Scheduling

- **Keep track of replicas**
  - o maintain file dependency graph
  - o co-locate all dependent/derived files

- **Co-schedule data and computation**

  e.g. by modifying PBS to ...
  - o schedule jobs to data
  - o replicate data to jobs
  - o predict future data usage
  - o prefetch data



Outline

- **MOTIVATION**
  - o What's a Grid? Why using Grids?

- **TWO TYPICAL APPLICATION DOMAINS**
  - o Computation Fluid Dynamics: FlowGrid
  - o Particle Physics: DataGrid

- **WHAT'S MISSING IN CURRENT GRID SYSTEMS**
  - o Fault tolerance
  - o Scalability
  - o Distributed data management
  - o Advance reservation, co-scheduling

- **CONCLUSION**

Conclusion

- **No future without Grid technology!**
  - more computation, more distribution, more collaboration

→ HU-Antrag Graduiertenkolleg

- **No future without autonomic computing!**
  - Build self-regulating systems.

- **“Data is the Grid’s Killer App!”**
  (Fran Berman, director SDSC and NPACI, 4/2002 at ICCS)

→ HU-Schwerpunkt
„Große Datenmengen im Web“