MIUR

Italian National Research Programme

Strategic Projects on Enabling Technologies for Information Society

FIRB

Grid.it :

a National Italian Project on Enabling Platforms for High-performance Computational Grids

Marco Vanneschi

Department of Computer Science, University of Pisa – vannesch@di.unipi.it

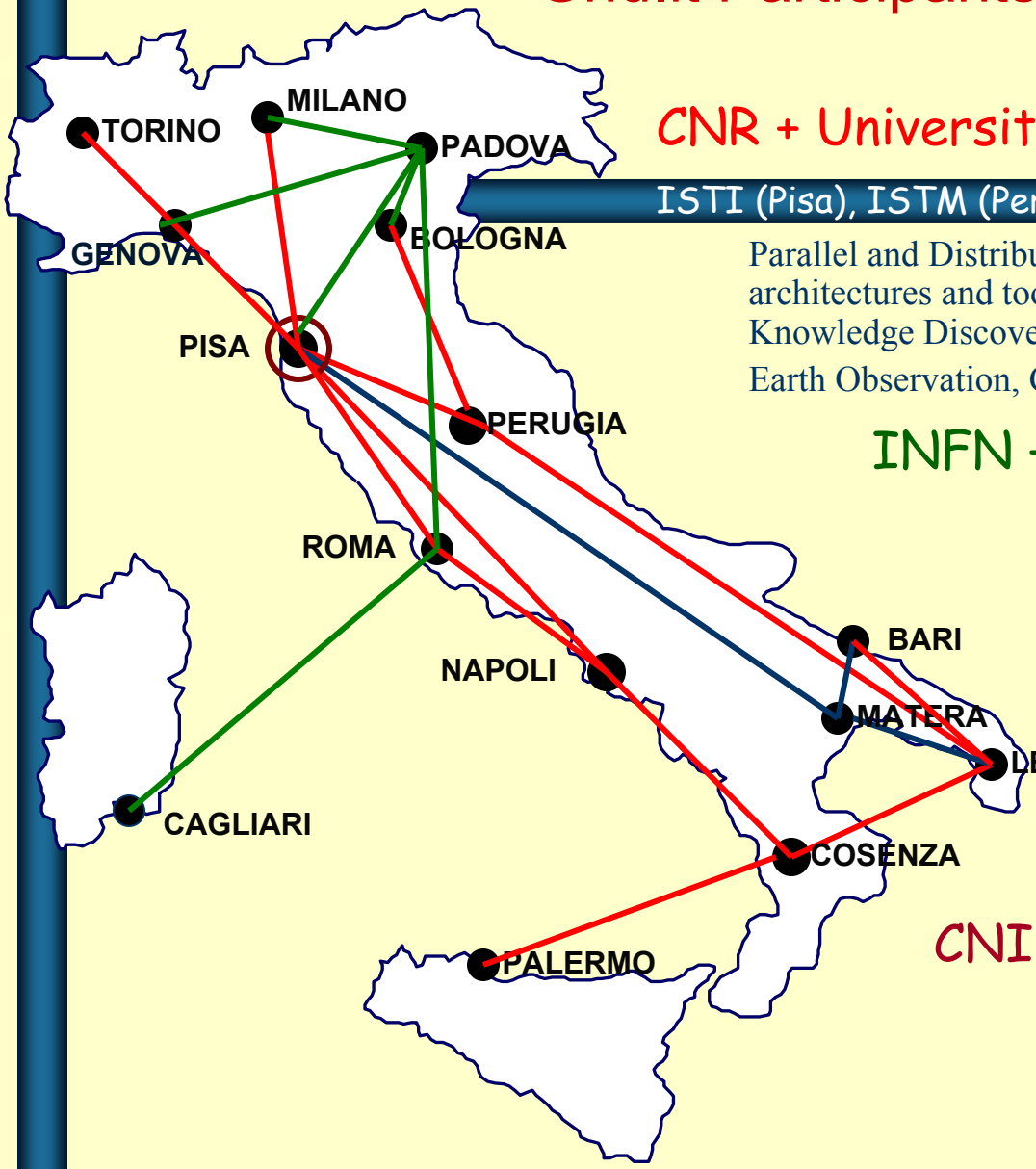
International Summer School on Grid Computing 2003, Naples

Grid.it Project

Enabling Platforms for High-performance Computational Grids Oriented to Scalable Virtual Organizations

- Basic Research Programme - ICT
  - + infrastructure and demonstrators (25%)
- Timeframe: November 2002 – October 2005
- Total Funding Budget (MIUR): 8,1 M€
  - of which 1,1 M€ for Young Researchers
  - total Cost: 11 M€
  - other synergies by MIUR-CNR Projects on Complex Enabling Platforms: 2,5 M€

Grid.it Participants (CNR project coordinator)



CNR + Universities (51%)

ISTI (Pisa), ISTM (Perugia), ICAR (Cosenza, Naples)

Parallel and Distributed Programming Environments, Grid architectures and tools, Scientific Libraries, Data Base and Knowledge Discovery,

Earth Observation, Computational Chemistry, Image Processing

INFN + Universities (20%)

Grid infrastructures

(INFN-Grid, DataGrid, DataTag) ,

e-science applications: Bioinformatics, Geophysics, Astrophysics,

ASI + Universities (11%)

Applications of Earth Observation

CNIT + Universities (19%)

Technologies and HW-SW infrastructures for high-performance communication, Optical technologies, …

Structure of Grid.it Project

Research Units:

1. CNR, ISTI
   Domenico Laforenza
2. CNR, ISTM
   Marzio Rosi
3. CNR, ICAR
   Almerico Murli
4. INFN
   Mirco Mazzucato
5. CNIT
   Giancarlo Prati
6. ASI
   Giovanni Milillo

Technical Board:

Coordinated by Domenico LAFORENZA

Research strategies to integrate the results of the various Workpackages (14).

Workpackages

WP1. Grid Oriented Optical Switching Paradigms (Castoldi, CNIT)

WP2. High Performance Photonic Testbed (Giordano, CNIT)

WP3. Grid Deployment (Mazzucato, INFN)

WP4. Security (Talamo, Univ. Roma Tor Vergata)

WP5. Data Intensive Core Services (Mazzucato, INFN)

WP6. Knowledge Services (Turini, Univ. Pisa)

WP7. Grid Portals (Aloisio, Univ. Lecce, ISUFI)

WP8. High-performance Component-based Programming Environment (Danelutto, Univ. Pisa)

WP9. Grid-enabled Scientific Libraries (Murli, Univ. Napoli & ICAR)

WP10. Grid Applications for Astrophysics (Benacchio, INAF)

WP11. Grid Applications for Earth Observation Systems Application (Milillo, ASI)

WP12. Grid Applications for Biology (Apostolico, Univ. Padova)

WP13. Grid Applications for Molecular Virtual Reality (Laganà, Univ. Perugia & ISTM)

WP14. Grid Applications for Geophysics (Navarra, INGV)

Structure of this presentation

- Part 1: General View of Grid.it Project

  Scientific Objectives and Topics

- Part 2: Grid.it Software Technology

  Programming Environment

Part 1

General View of Grid.it Project:

Scientific Objectives and Topics

Basic research objectives

- General strategic ICT objective:
  - overcome (some) current limitations in Grid architecture and in environments / tools for application development,
  - for new Grid platforms being much more pervasive and oriented to the user requirements,
  - yet compliant with standardization efforts (OGSA) and open source requirements.
- Current version of Grid.it software technology:
  - on top of (a subset of) Globus,
  - for the most part, the highest levels are new.

Specific research objectives

Software technology of Grid.it :

- High-level programming environment
- Knowledge services
- Scientific libraries
- Resource management
- Security
- Support to Virtual Organizations

Software technology of Grid.it

Domain-specific Problem Solving Environments (PSEs)

High-level services

Knowledge services, Data bases, Scientific libraries, Image processing, …

High-performance, Grid-aware component-based programming model and tools

Programming Environment

Resource management, Performance tools, Security, VO, …

Next Generation Middleware

Basic infrastructure, Globus-compliant

Grid programming environment

- High-level tools
  - Better programmability and productivity
  - Effective software reuse, including legacy
  - *Grid-aware* : dynamic context and adaptive applications
  - Performance prediction and modeling
- High-performance, Grid-aware component technology
  - High-level models and tools for high-performance, *adaptive*, structured composition of Grid applications

New middleware

- *Light* (Risc-like) approach to middleware core services
  - OGSA compliant
- Core services definition and realization: mainly according to the needs of the programming environment
  - Resource management and discovery
  - Performance modeling
  - Virtual Organizations
  - Certification and Security
    - Mechanisms able to overcome the PKI limitations

Knowledge services

- Knowledge-intensive applications and processes
  - Information extraction and Knowledge Discovery (Data Mining) from structured and semi-structured sources
  - High-performance Search, Query Answering, and Retrieval Services
  - Grid-aware Data Base and Information Systems
- Use of extracted information and knowledge to assist the *resource management and discovery* tools in the programming environment support

Infrastructures

- Globus-based production Grid (INFN)
  - Tools for deployment, management and monitoring
  - Data intensive services
  - Basic support to the highest levels of Grid.it software technology and application development
- High-performance photonic testbed (CNIT)
  - High-performance communication services in Metropolitan Area Networks belonging to the national backbone (GARR)

Grid-aware demonstrators

- Grid applications for
  - Biology
  - Earth Observation Systems
  - Molecular Virtual Reality
  - Geophysics
  - Astrophysics
- Testbeds for the Grid.it software technology

Part 2

Grid.it Software Technology:

Programming Environment

Grid.it Programming Environment

- Development environment for Grid-aware applications
  - Heterogeneous, dynamic, *adaptive* context
  - Grant a certain degree of QoS: performance, fault tolerance, security
- High-level environment, tools and methodology: the programmer has a very abstract view of the Grid
  - Resource management and service utilization: mainly at the responsibility of the environment tools
- High-performance
  - Grid-computing *vs* parallel-distributed computing
  - methodologies and technologies: new *vs* revisited

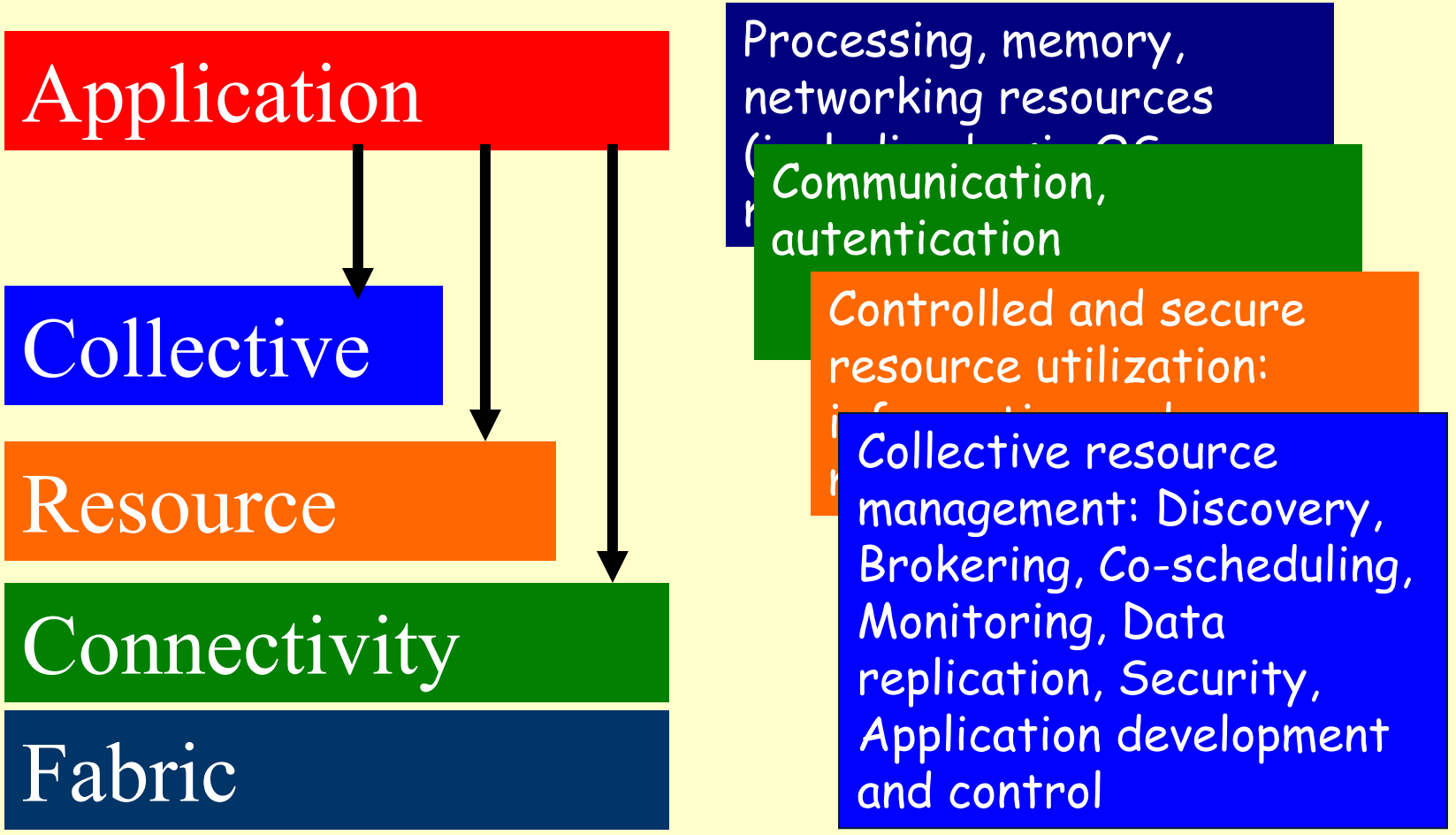
Grid platforms

- "Distributed computing infrastructure for coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations"

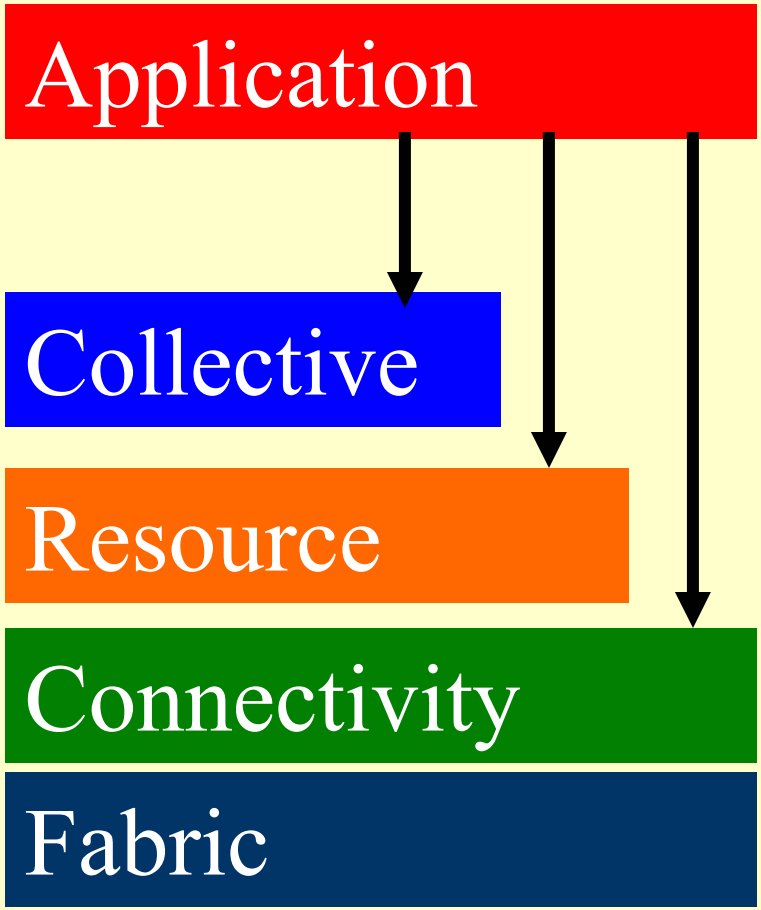
- w.r.t. distributed-parallel platforms:

  an advancement, not a replacement

Current view of Grid applications (1)



Current view of Grid applications (2)



Conceptually, nothing new w.r.t. programming on top of OS services and mechanisms, …

… but much more difficult for Grids !

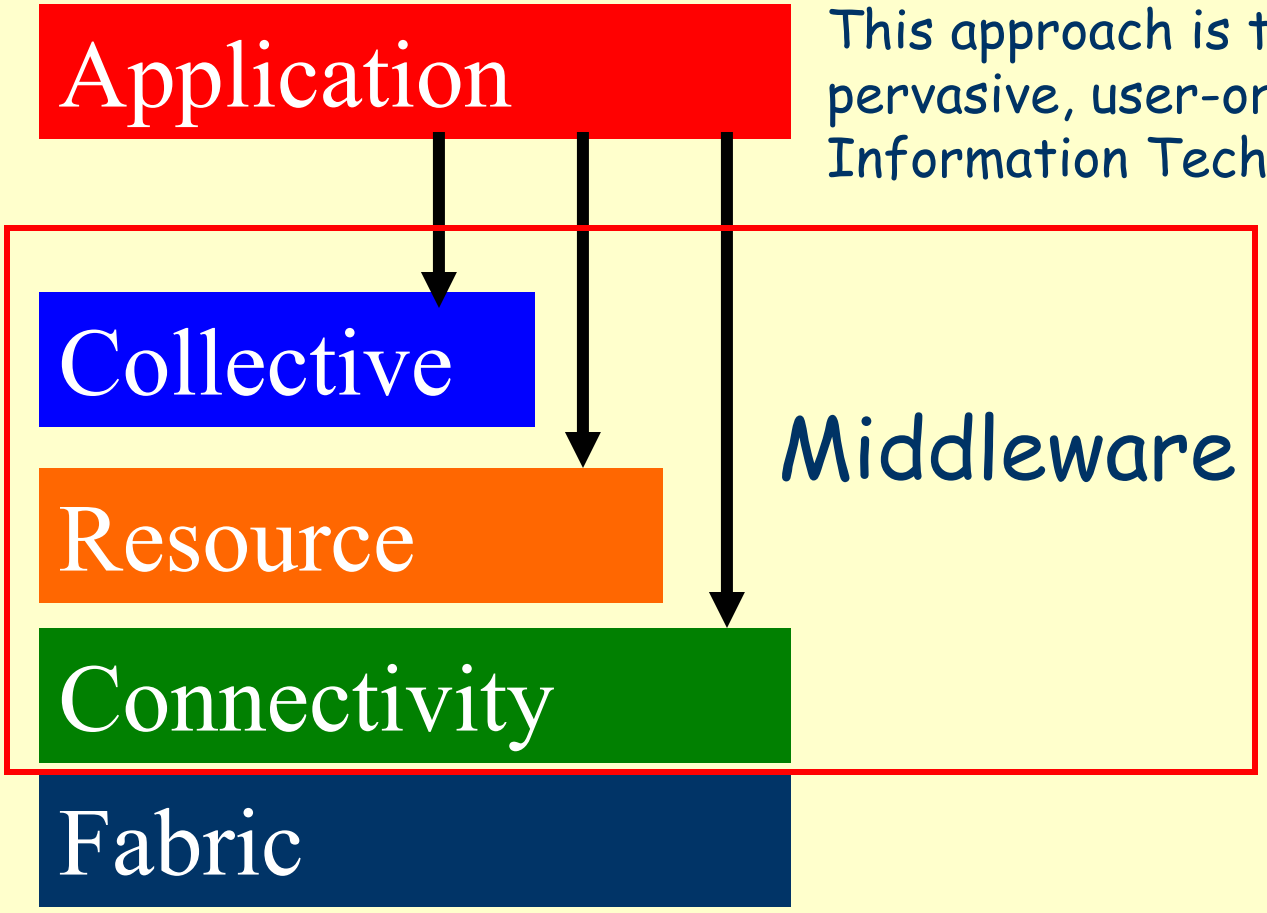
Application management: collective resource control, dynamic allocation

Resource management by process servers and their composition

(Micro-)kernel: interprocess communication, protection, …

Basic HW + SW

Current view of Grid applications (3)



This approach is too low-level for a pervasive, user-oriented Information Technology.

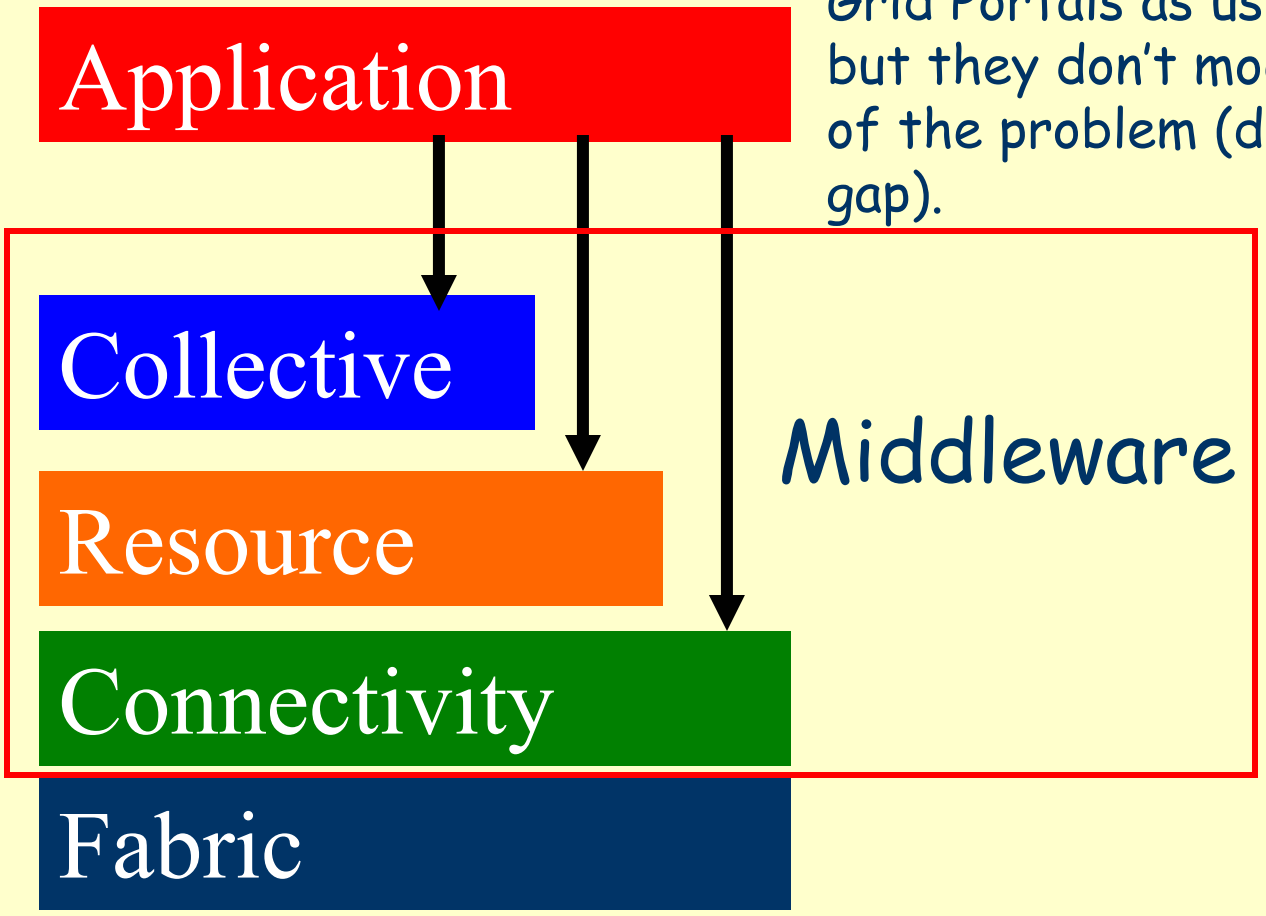
Programming directly on top of services

is hard, expensive, error-prone, …

has low productivity, …

… and much more difficult for Grids !

Current view of Grid applications (4)



Grid Portals as useful in this view, but they don't modify the nature of the problem (don't eliminate the gap).

Applications as collection of services,

to be composed, optimized, accessed, controlled, … *directly* by the application designer

High-level view of Grid applications

- High-level languages, compositionality, modularity and interoperability
- Compiling Tools
- Run Time Support
- Programming Model (Cost Model) for static and dynamic optimizations
- Development, loading, execution, monitoring,…, reconfiguring tools

It is not necessarily the same Middelware as before: it should be defined and realized according to the needs of the Programming Environment

Current *vs* high-level view: reasons for the gap

- Non conventional problems
  - **heterogeneous**, **dynamic**, **adaptive** applications
  - **new cost models** *w.r.t.* conventional and homogeneous parallel systems
- Web-like view of Grid
  - Interoperability, utilization of existing mechanisms, on-the-fly tech, …
  - Instead, we need to move towards a **complete virtualization of Grid resources** – at the same time preserving successfull Web mechanisms
- Needs for a new high-level Grid software technology
  - An outstanding R&D challenge
  - *Exploiting all the past experiences* (parallel computing, software engineering, …)

Current mechanisms

- Message-passing (MPICH-G)
- RPC / RMI
  - e.g. NINF-RPC
- From Web services to OGSA services
  - even in this case, the programmer interacts directly with the Middelware
- All these mechanisms are still too low-level for the modular, robust development of complex Grid-aware applications
  - Some of them could be exploited at the implementation level of the Programming Environment (i.e. the importance of standards is preserved)
- Often, performance / scalability is a seriuos problem too
  - more efficient Middleware mechanisms (Risc-like ?) and optimizations are needed

Example of innovation: GrADS Project

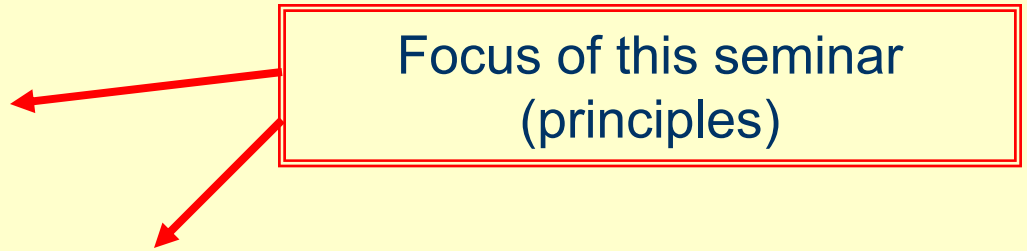
- Concept of reconfigurable program
  - High-level formalism
  - High-level information on application requirements
  - Components technology and composition of applications
  - Performance model ("negotiation" at run-time)
- Application manager:
  - set of static and *dynamic* tools that control all the development-execution cycle of the application (including dynamic restructuring)

Grid.it and Programming Environment

- Workpackage 8 (WP8) : Programming Environment
  - Coordinator: Marco Danelutto, Department of Computer Science, University of Pisa
  - Universities of Pisa, Naples, Cosenza, Milan; CNR institutes in Pisa, Naples, Cosenza, Palermo, Genova, Roma
- WP8 is central to Grid.it
  - Basic approach to the research of the whole Project
  - Strong coordination with other WP in Grid Software Technology
    - WP4 (Security), WP6 (Knowledge Services), WP7 (Grid Portals), WP9 (Scientific Libraries)
  - and with Applications WPs and Networking WPs.

Critical research issues

- Dealing with heterogeneity
- New compilers, run-time supports
- Secure and fault tolerant implementations
- Dynamic, adaptive applications
- Implementing requirements for Quality of Service



Grid.it approach

- Coordination language
- Performance model
- High-performance component technology
- Resource management services integrated in the Run-time of Programming Environment

Grid.it approach : coordination language and cost model

- Our past experience in parallel programming environments
- Skeletons model $\Rightarrow$ Structured Parallel Programming
  - high-level constructs for task parallelism (e.g. PIPELINE, FARM), data parallelism (e.g. MAP, STENCILS), mixed task+data parallelism (PARMOD), and their compositions (GENERIC or STRUCTURED GRAPHS)
- Semantic model and associated performance model
  - constraints on the parallel paradigm adopted to compose (sequential / parallel) modules into complex applications
- Many potentialities for intensive optimizations and restructuring of applications

Grids and parallelism : why ?

- Applications may contain parallel components
  - in the simplest case, a parallel component is allocated to a single node (cluster, supercomputer),
  - advancement in networking technology: parallelism can be effectively exploited at the large-scale level too.

- More in general, and more important: *structured* parallelism is a methodology to design and to manage high-performance, *Grid-aware* application components according to QoS requirements.

Structured Parallel Programming

ASSIST

A Programming Environment for High-performance Portable Applications on Large-scale Platforms

Projects:

- ASI-PQE2000
- CNR Agenzia 2000
- MIUR-CNR Strategic Programme L449/97, 1999 e 2000
- MIUR-FIRB Grid.it

Implementations:

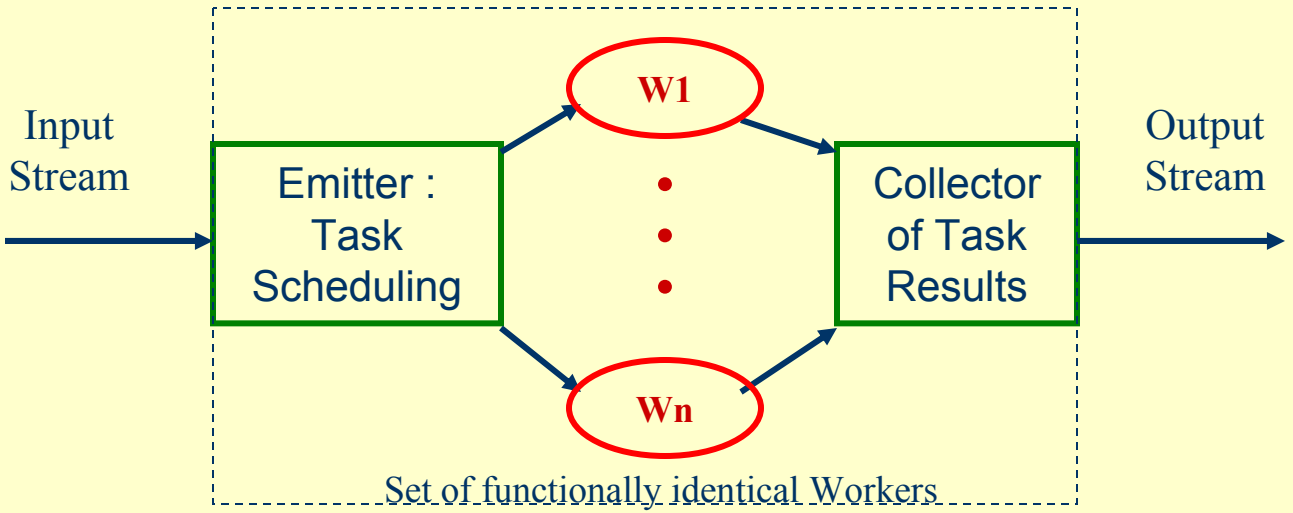
- **Cluster/Beowulf** (on top of ACE)
- **First Grid version – AssistConf** (on top of Globus)
- On-going: **Component Assist for Grid.it**

http://www.di.unipi.it/research/TR

Structured parallel programming and performance models (1)

Example: Farm / Master-Slave / Parameter Sweeping / …

Load-balanced execution of Tasks belonging to a Stream



Efficient *implementation templates*

for platform- and application-dependent optimizations

Optimal number of workers and other performance parameters (e.g. throughput, efficiency) can be expressed as functions of processing times, communication times, and utilization factors

Structured parallel programming and performance models (2)

- **Performance models and implementation templates** can be defined for all the most common task-parallel (pipeline, farm, ..) and data-parallel (map, reduce, parallel prefix, fixed stencils, variable stencils, …) and for many compositions of them
- First experience: P^3L / SkIE
- ASSIST performance model and templates for generic graphs and generic skeletons (parmod) + external objects

Software technology of Grid.it

Domain-specific PSEs

High-level services

Knowledge services, Data bases, Scientific libraries, Image processing, …

High-performance, Grid-aware component-based programming model and tools

Programming Environment

Resource management, Performance tools, Security, VO, …

Next Generation Middleware

Basic infrastructure, Globus-compliant

High-performance, Grid-aware component technology

- **Components model** is recognized as a valid technology for compositionality, interoperability, software reuse, application versioning

- However, **high-performance, Grid-aware** components are needed
  - Current component technology, derived by Object technology, is not efficient in complex applications, and it is not suitable for Grid-aware applications
  - Current research projects: CCA, XCAT, CCM, … (not all are for Grid)
  - A notable attempt: **GrADS** Project

Grid.it approach:
high-performance, Grid-aware component technology

- **Joining component technology** and **structured parallel programming technology** to achieve the goal of **high-performance, Grid-aware, component-based applications**
- **This is the intimate link between Grid programming and parallel programming**
  - **In the simplest cases, structured parallelism is exploited inside Grid nodes (clusters, supercomputers)**
  - **Any way, structured parallelism is exploited at Grid-wide level in order to express Grid-aware applications**
    - **Dynamic, adaptive, QoS**

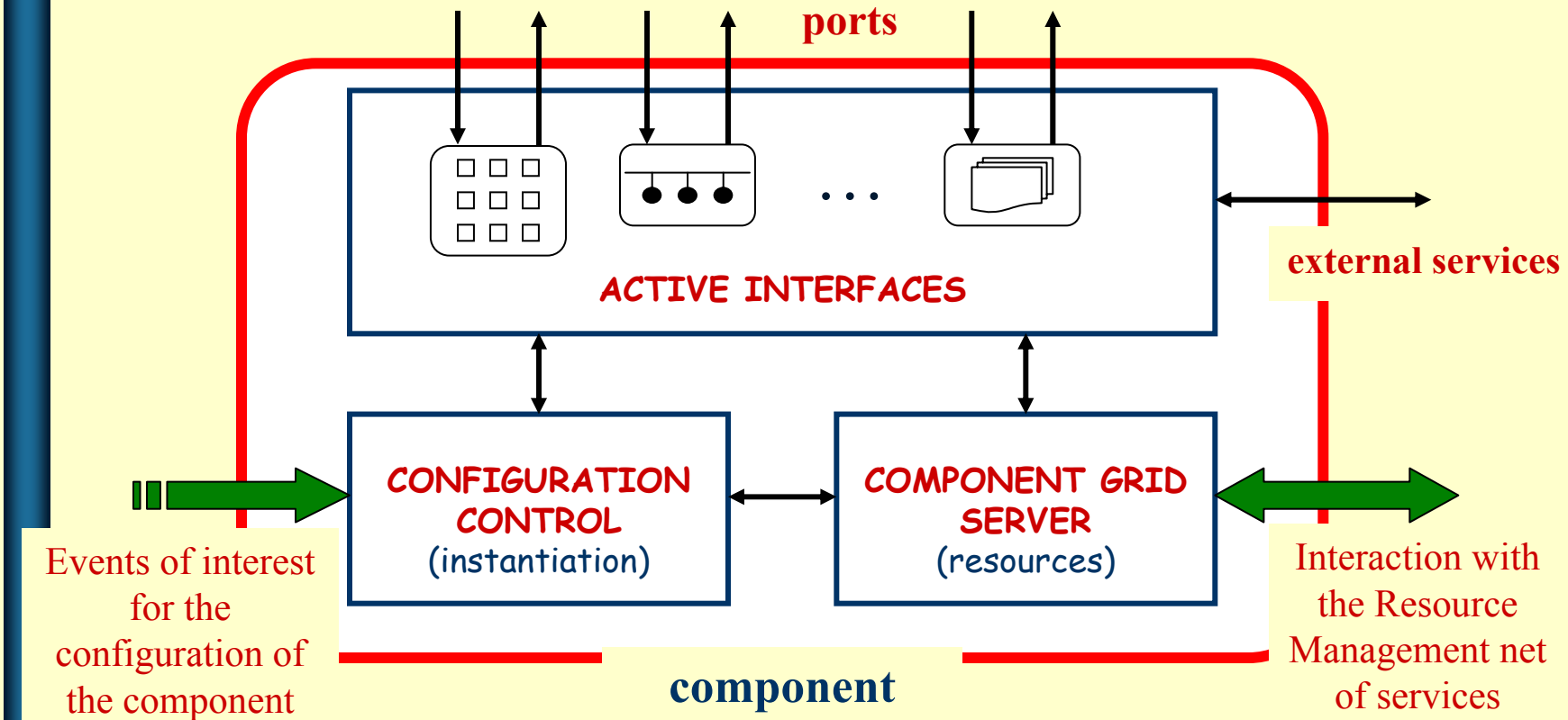
Grid.it approach : performance model and QoS

- Proper (re-)structuring of applications, acting on
  - Distribution / parallelism
  - Data management, and so on …
- Dynamically modifying the allocation, replication / partitioning of the application components, in order to mantain the proper degree of performance, or in order to significantly increase performance when necessary
- Dynamic use of the performance models and implementation templates (components + structured parallel programming)

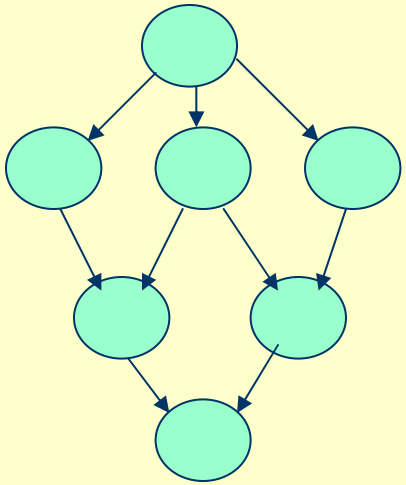
Grid.it approach : basic ideas for Grid-aware components

- "**Contract**" associated to every component *interface*, defining the possible application requirements:
  - Performance,
  - Fault tolerance, …
- Every contract is specified according to a parallel-distributed program
  - e.g. using the ASSIST model
- An initial configuration of the program is established at compile-time, according to the cost model of the composition of components
- At run-time, the cost model is used to modify the configuration of the composition (in a parametric manner):
  - *replication*, *partitioning*, *scheduling policy*, *distribution of data*, … (all are *constructs* in ASSIST and other structured models)

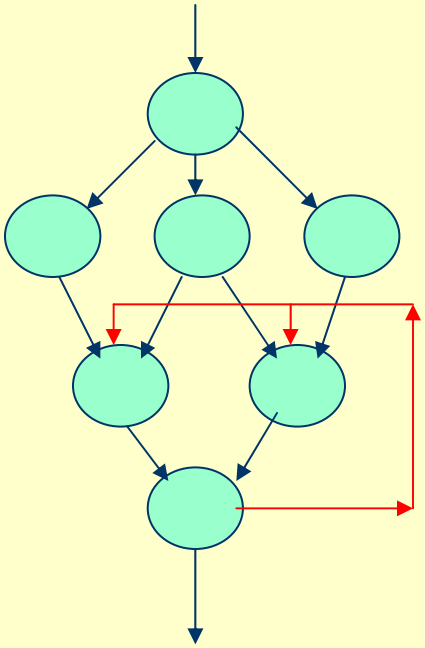
Grid.it approach: “Active Interfaces” implementation model of high-performance adaptive components



Examples of structures for Grid application



Acyclic precedence graph (DAG) of components with data-flow behaviour. (Workflow)



Stream-based, possibly cyclic graph of components : data-flow and/or nondeterministic behaviour

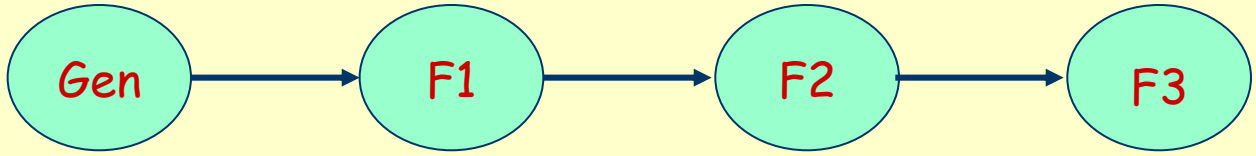
In both cases, nodes (=components) can be expressed by parallel constructs.

A global performance model (e.g. queueing network), based upon the performance models of the parallel constructs, can be applied dynamically.

Information for the performance model are acquired by monitoring, profiling, …

The application may be restructured at run-time.

Example: an “adaptive pipeline”



Generator of objects stream

Objects transformation by function F1

Objects transformation by function F2

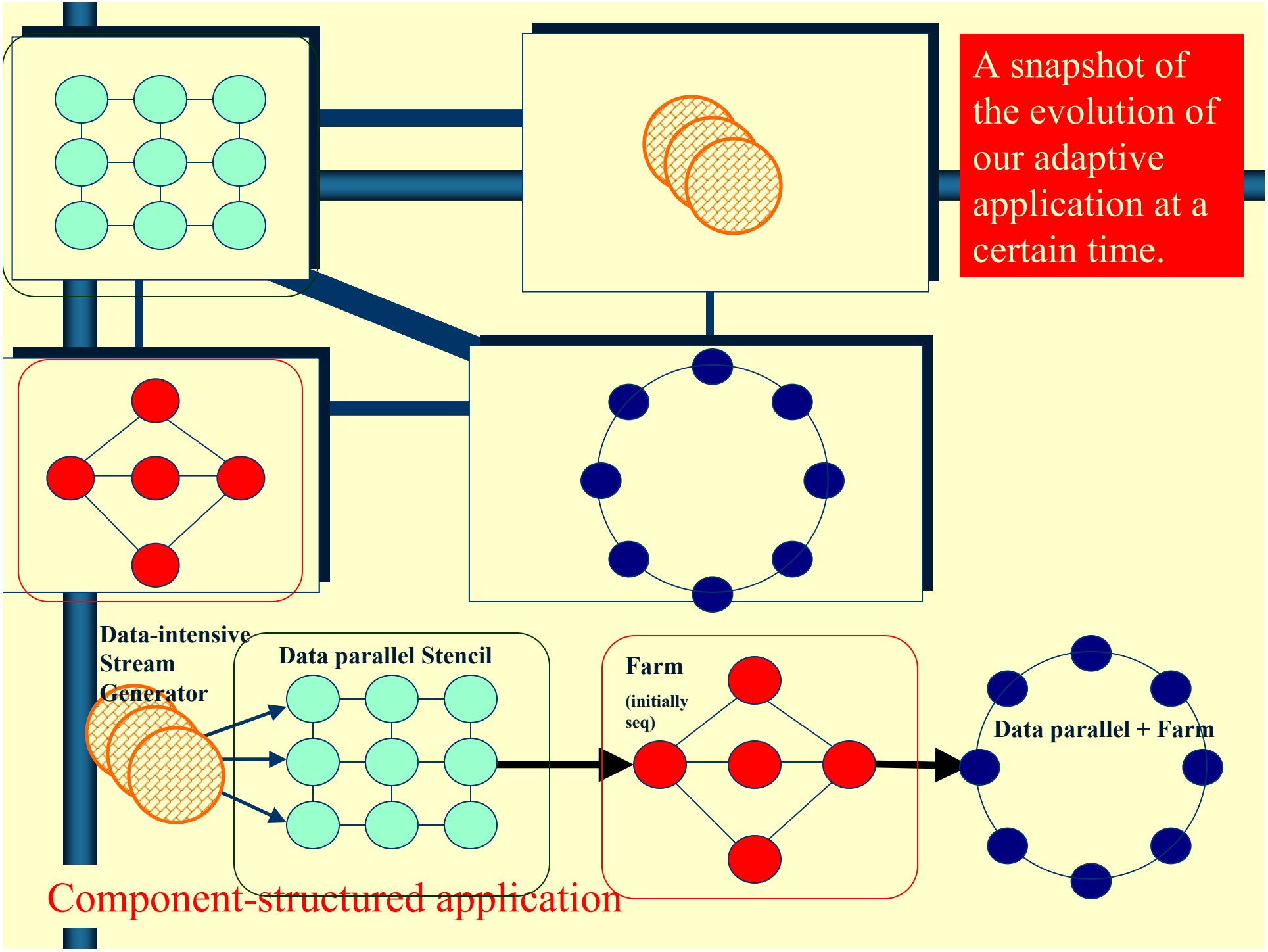
Objects transformation by function F3

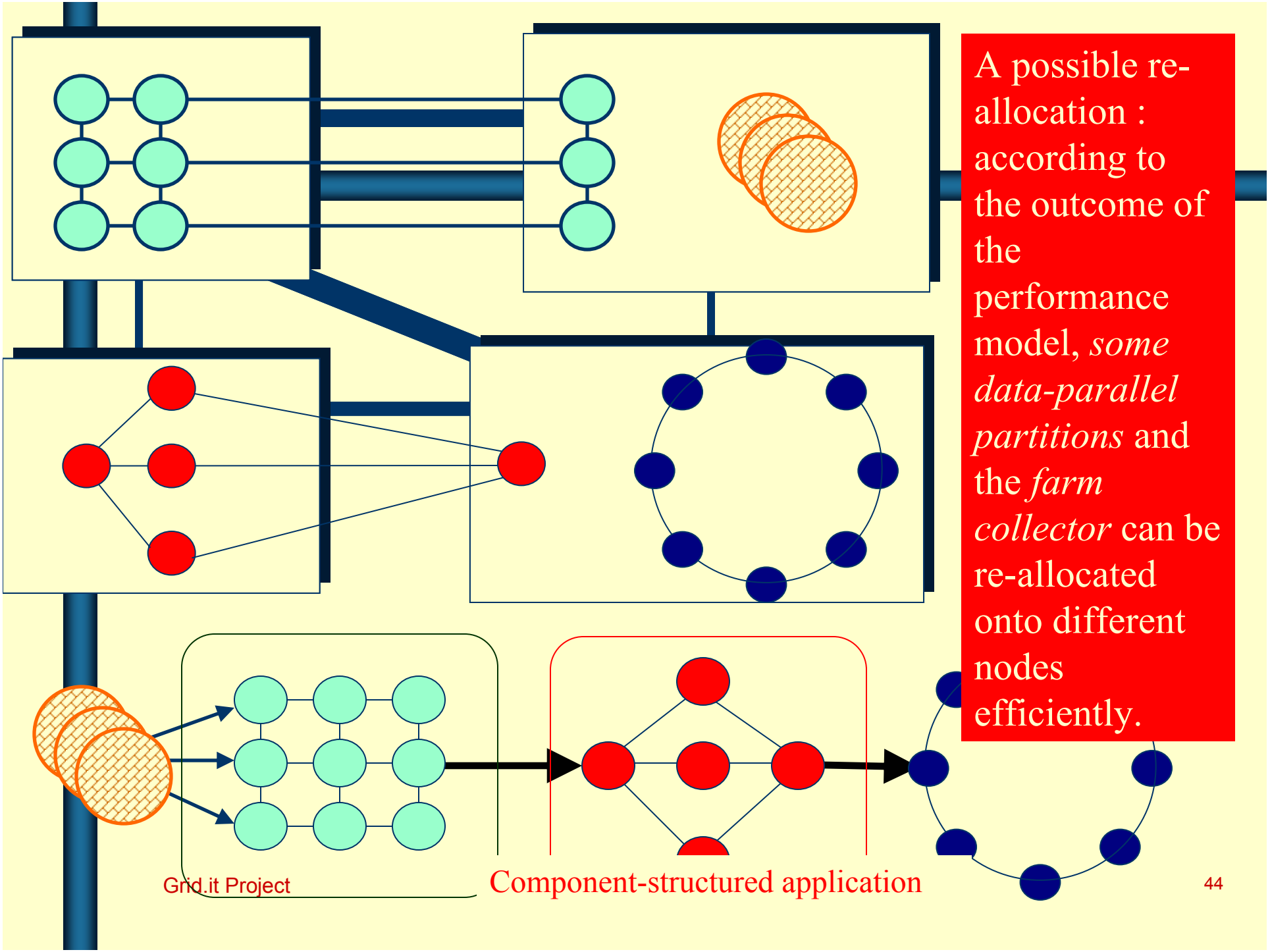
Data intensive, memory hierarchy on Grid.

- By default: **data-parallel** implementation onto a **single parallel node**.
- On restructuring: the number of partitions may be varied and allocated onto different nodes.

- By default: **sequential** implementation.
- On restructuring: **farm** implementation,
- number of workers determined dynamically.

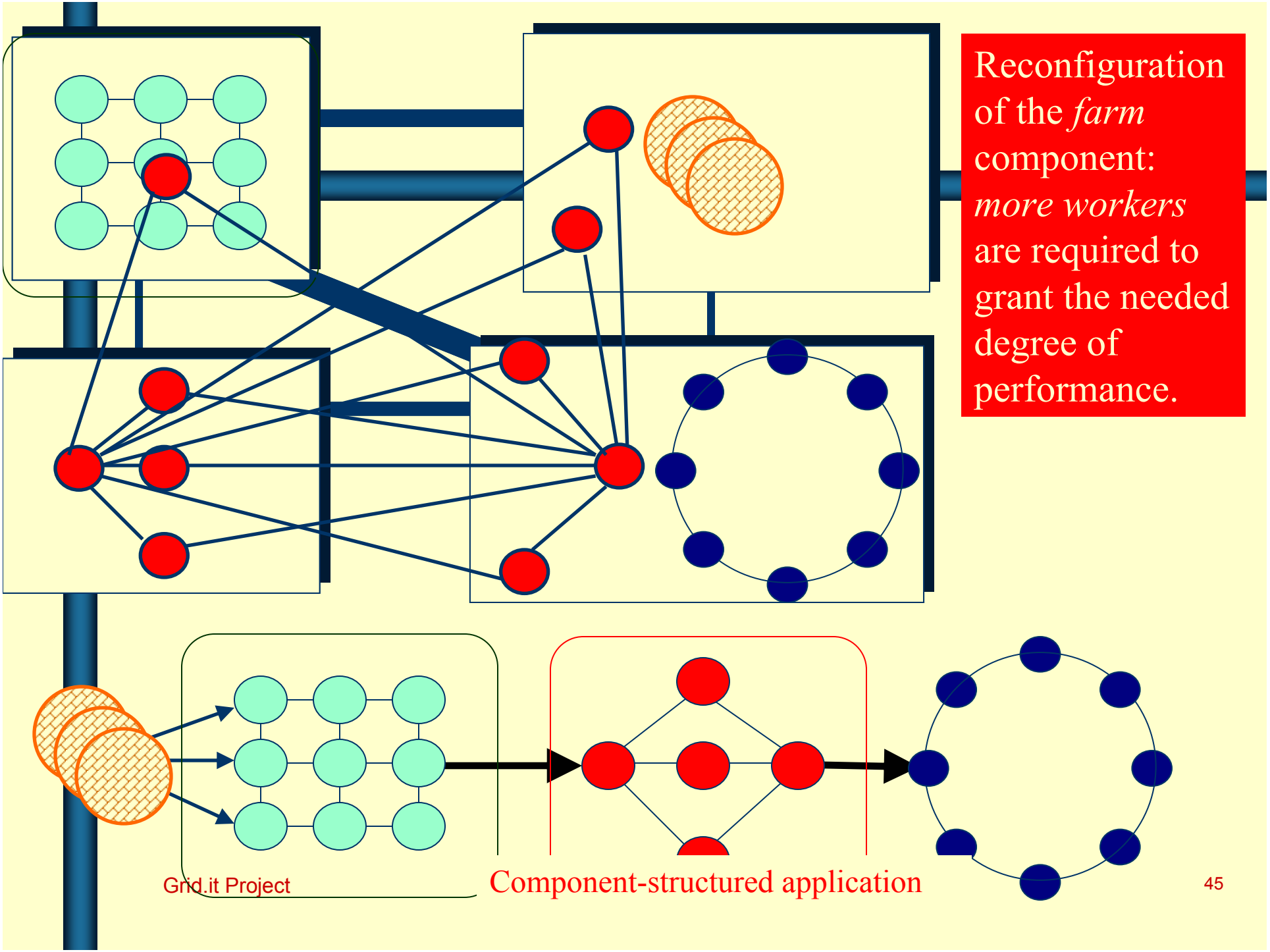
A stream parallel + data parallel composition mapped onto **a single parallel node**.


A snapshot of the evolution of our adaptive application at a certain time.
Data-intensive Stream Generator
Data parallel Stencil
Farm
(initially seq)
Data parallel + Farm
Component-structured application

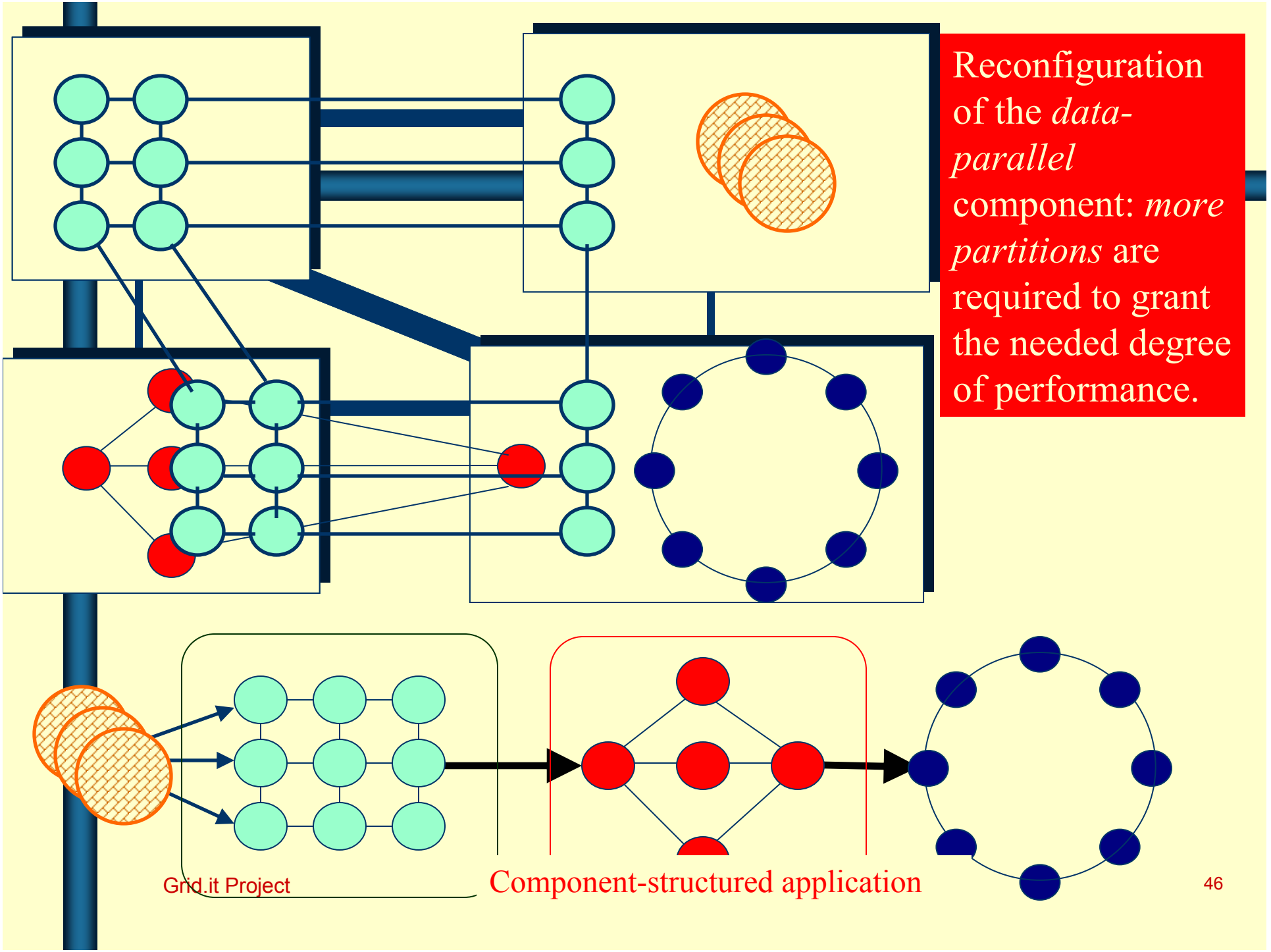


A possible re-allocation : according to the outcome of the performance model, *some data-parallel partitions* and the *farm collector* can be re-allocated onto different nodes efficiently.

Component-structured application

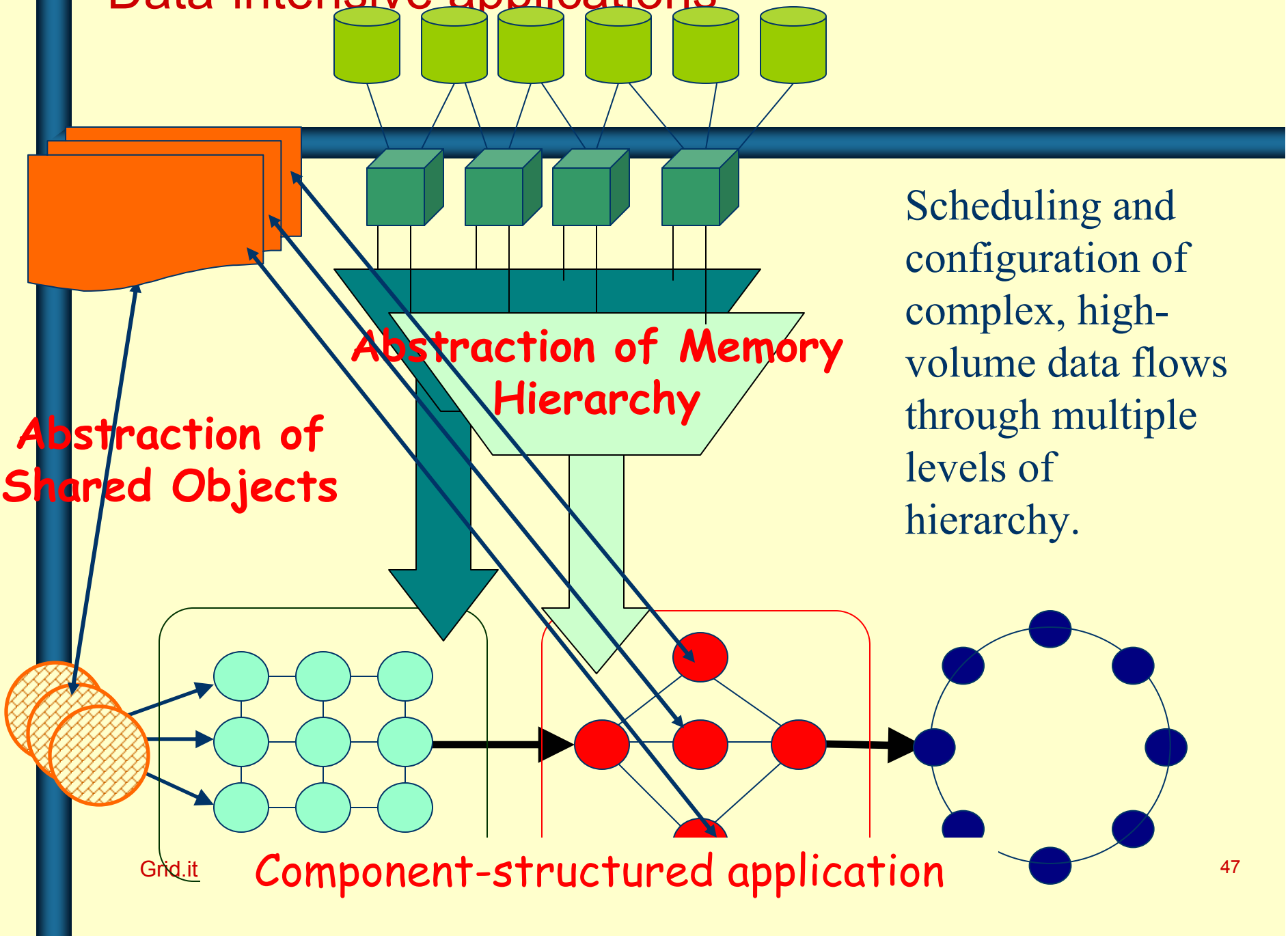


Component-structured application



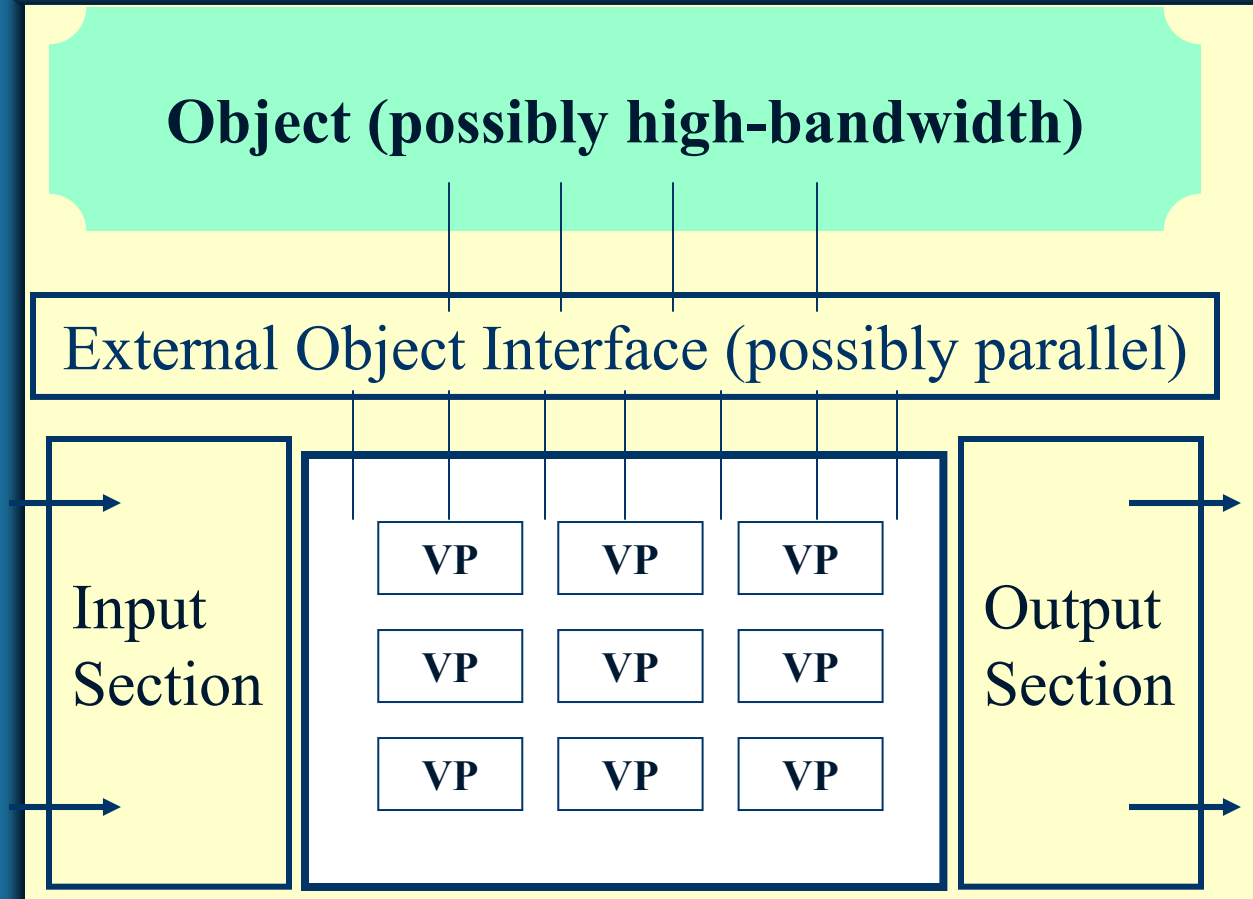
Component-structured application

Data-intensive applications



Scheduling and configuration of complex, high-volume data flows through multiple levels of hierarchy.

Data-intensive computations in ASSIST



ASSIST parmod: high-performance abstraction of Object

Abstraction of high-performance objects can be implemented by ASSIST *parmod*(s),

with proper interface (expressed in ASSIST or another formalism)

Grid.it approach : components and programming levels

- More than one level of components ?
  - Components for the User level
  - Components for the Specification / implementation of requirements
  - Components for the Run-time support
    - Modular, robust exploitation of underlying (e.g. Globus) services (component themselves)
- Unified methodology (high-performance components) with different instantiations where necessary
  - Efficiency
  - Expressive power, …

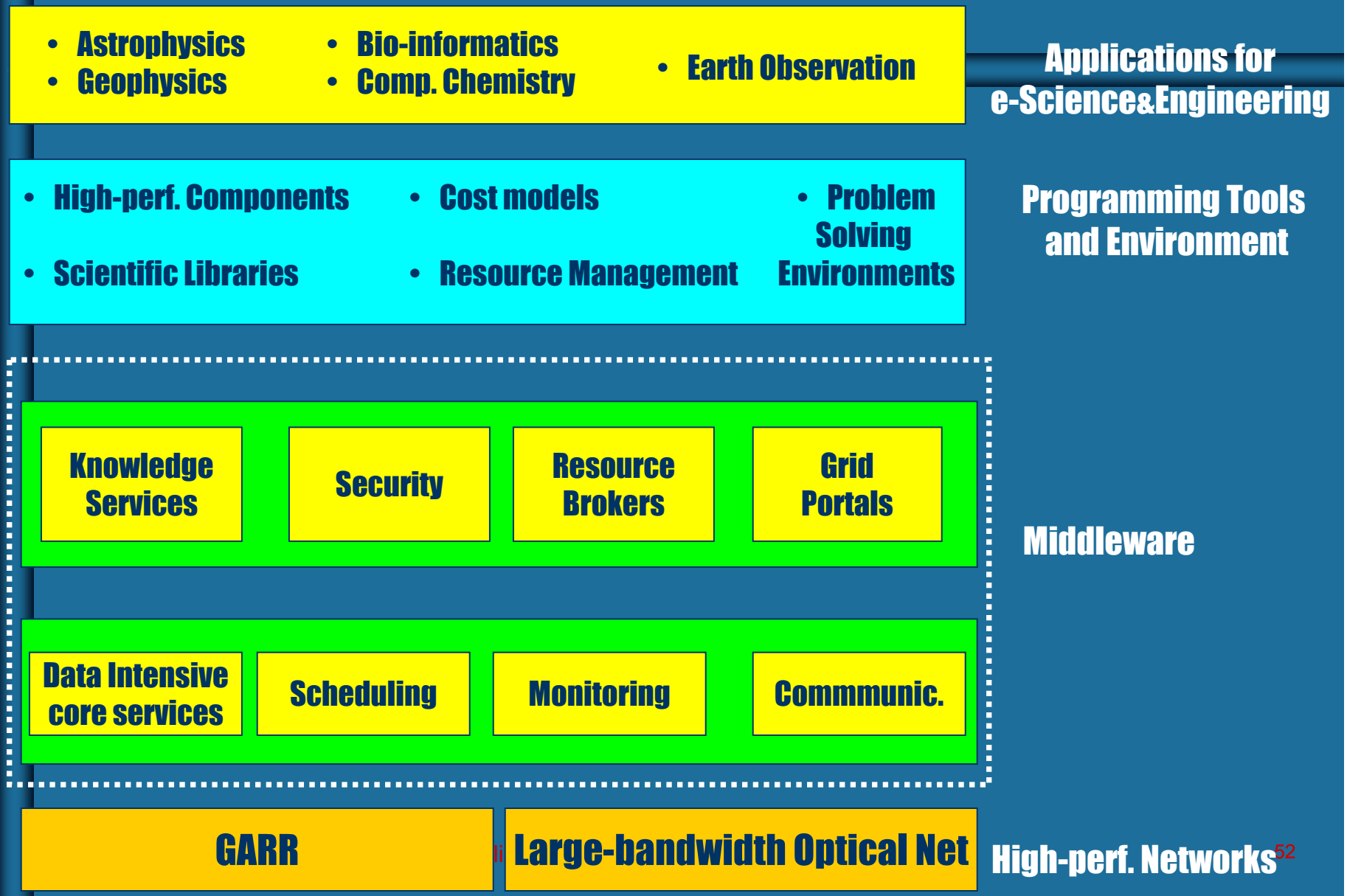
Thanks to many Grid.it people

In particular:

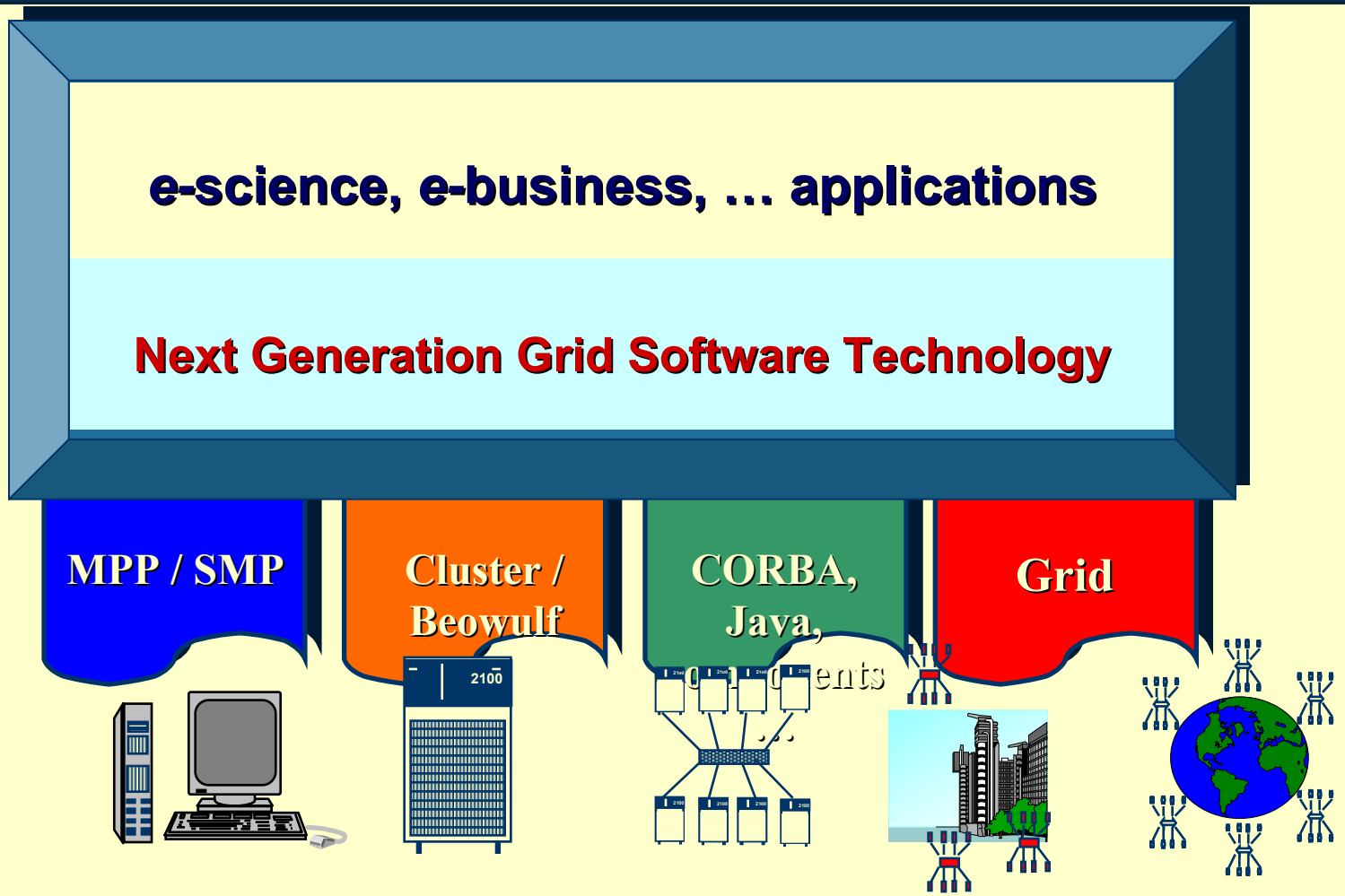
- **Marco Danelutto**, Marco Aldinucci, Massimo Coppola, Paolo Pesciullesi, Massimo Torquati, Corrado Zoccolo
  - Department of Computer Science, University of Pisa
- Domenico Laforenza
  - ISTI-CNR, Pisa
- Salvatore Orlando
  - Department of Computer Science, University of Venice

Thank you for attention

Research activities and system levels

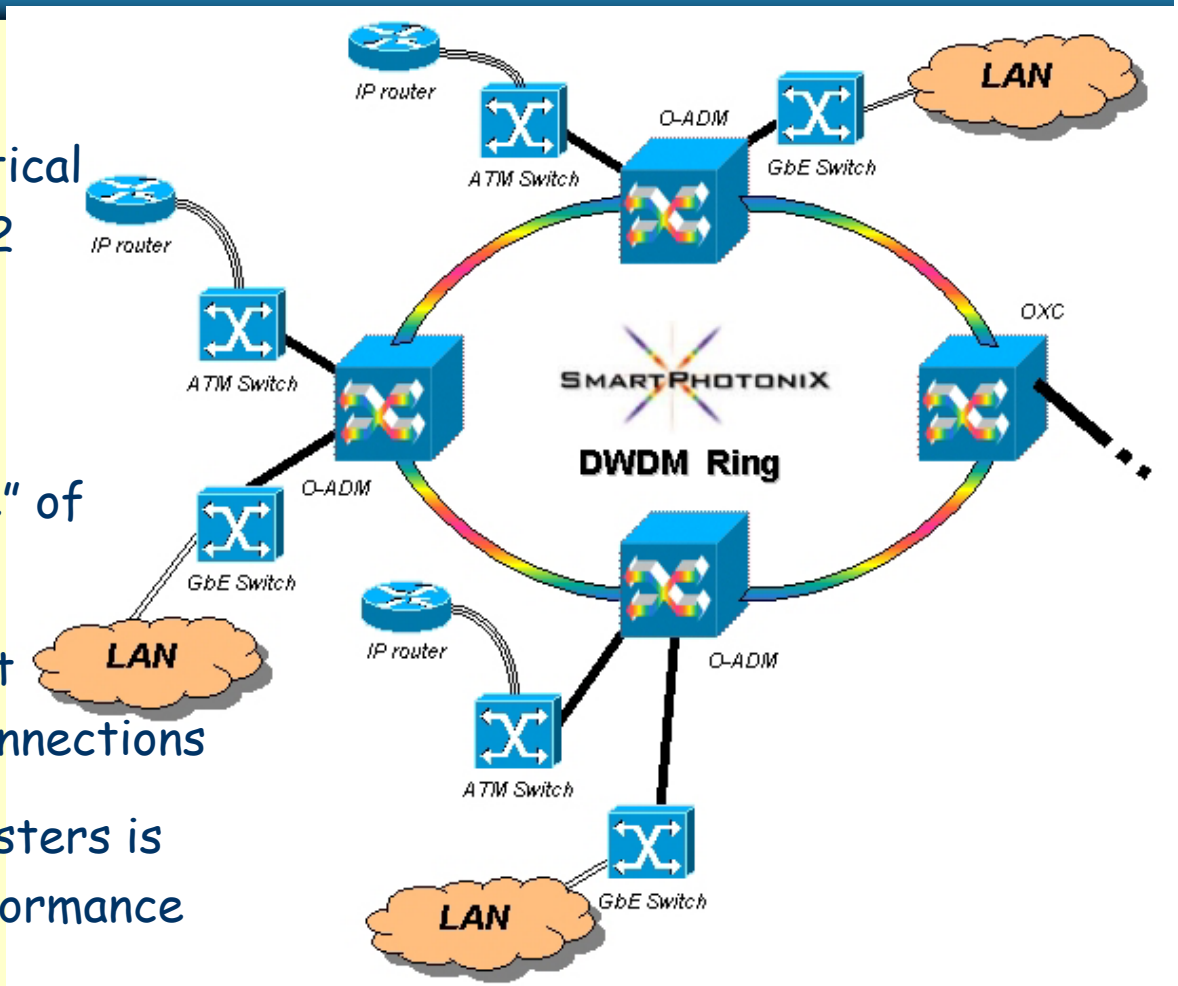


Seamless High Performance Computing



High-performance networks

- Very large bandwidth optical fibers (DWDM 10Gb x 32 channels): "Metro-core"
- MAN ring (Pisa)
- High-performance "Node" of GARR Network
- Cell (IP/ATM) and packet (IP/Gigabit Ethernet) connections
- A collection of LANs/clusters is seen as a same high-performance machine



Grid.it:
new middleware as Grid Abstract Machine

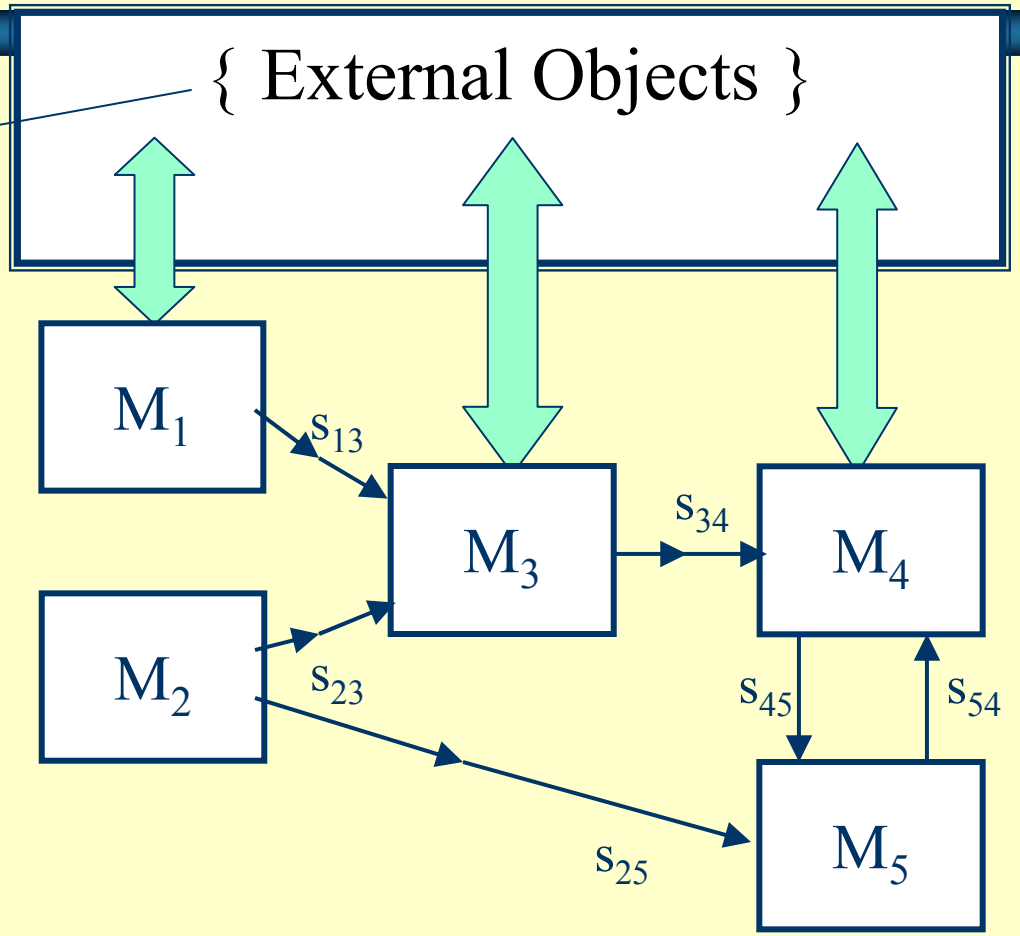
- *Light* (Risc-like) approach to middleware core services
  - OGSA compliant
- Core services definition and realization: mainly according to the needs of the programming environment
  - Resource management and discovery
  - Performance modeling
  - Virtual Organizations
  - Certification and Security
    - Mechanisms able to overcome the PKI limitations

ASSIST Coordination Language (ASSIST-CL)

- Programs as generic **graphs** whose nodes are parallel modules and/or sequential modules
  - Compositionality through streams
  - Data-flow behaviour + state + nondeterminism
  - Programs reusable as nodes of other programs
- Programs can also be expressed as structured graphs of basic skeletons (***pipeline, farm, loop*** skeletons), where the composed modules are ASSIST-modules

ASSIST: application structuring through Parallel Modules and Objects

- Global variables
- Shared memory
- **Files and I/O**
- Web
- **CORBA objects**
- **Agents (e.g. NetSolve)**
- ASSIST modules



General paradigm for parallel components: Parallel Module **(parmod)**