Integration Workshop
Krakow, Poland
October 19, 2006

CoreGRID

Fault-tolerant data sharing for high-level grid programming: a hierarchical storage architecture

Marco Aldinucci, Marco Danelutto
Dept. of Computer Science, University of Pisa, Italy

IN SUPREMÆ DIGNITATIS 1343

Gabriel Antoniu, Mathieu Jan
INRIA Rennes, France

INRIA RENNES

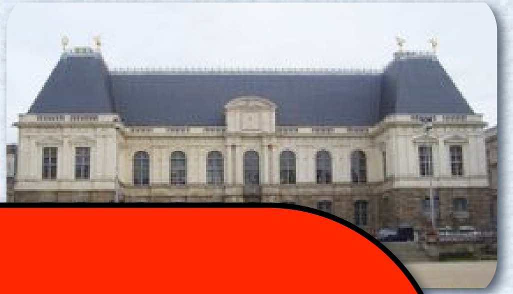
IRISA Institut de Recherche en Informatique et Systèmes Aleatoires

Marian's exercise ...

UniPisa

Grid-enabled high-level programming model (with data sharing)

INRIA Rennes



Robust data sharing service for the grid

Marian's exercise ...

UniPisa

INRIA Rennes

ASSIST with its cluster-oriented sharing service

JuxMem Jxta-based fault-tolerant sharing service

Memory hierarchy transparently supporting grid-level coherent, fault-tolerant, persistent data sharing. First prototype supports data sharing in ASSIST applications

Marian's exercise ...

UniPisa

ASSIST with its cluster-oriented sharing service

... while waiting for the "Philosophy of the Grid"

Not innovative?

At the bottom line, Grid appears more evolutionary than revolutionary, isn't it?

Memory hierarchy transparently supporting grid-level coherent, fault-tolerant, persistent data sharing. First prototype supports data sharing in ASSIST applications

Outline

- The two software tools
  - ASSIST (high-level programming model)
  - JuxMem (grid data service)
  - exploit their complementarity
- How they have been integrated
  - a memory hierarchy, with locality
- Prototype, experiments (preliminary)

Data management in grid

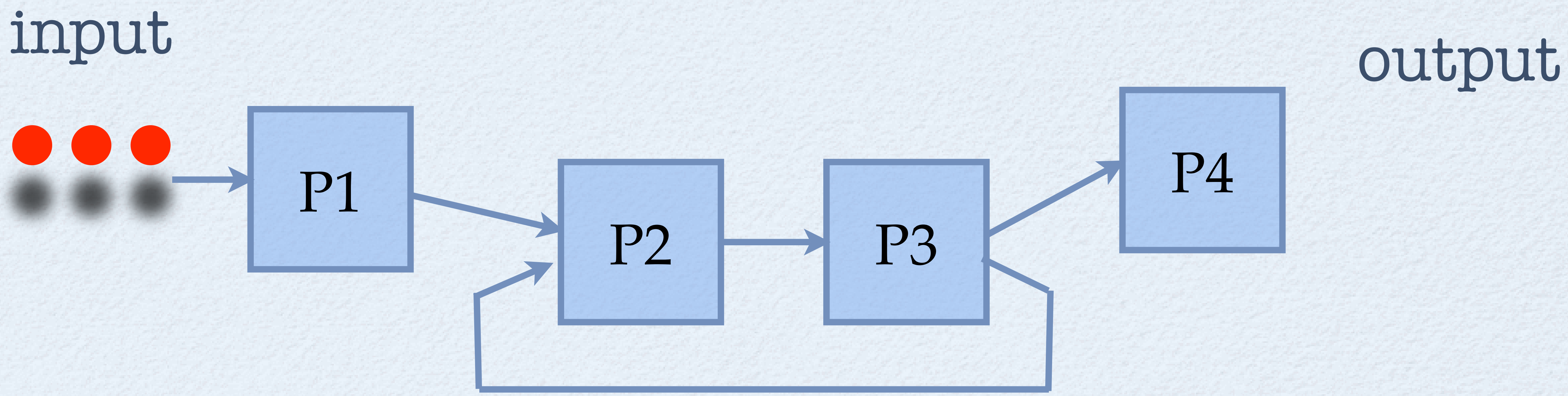
- Memory storage features
  - Persistency (survive to application instances)
  - Robustness (fault-tolerance)
  - Efficiency (not only in ftp, but real RAM storage)
- In high-level programming models
  - Transparent access from programming model
  - Run-time supp. implementation (e.g. FT message logs)

ASSIST @ UniPisa

- High-level programming model
- Based on parallel modules
  - GCM components ongoing
- Modules exchange data via streams **and/or** shared memory
  - sharing implemented via distributed memory server (called ASSIST/ad-HOC)
  - read, write (in parallel) distributed *"objects"* identified by a logical ID

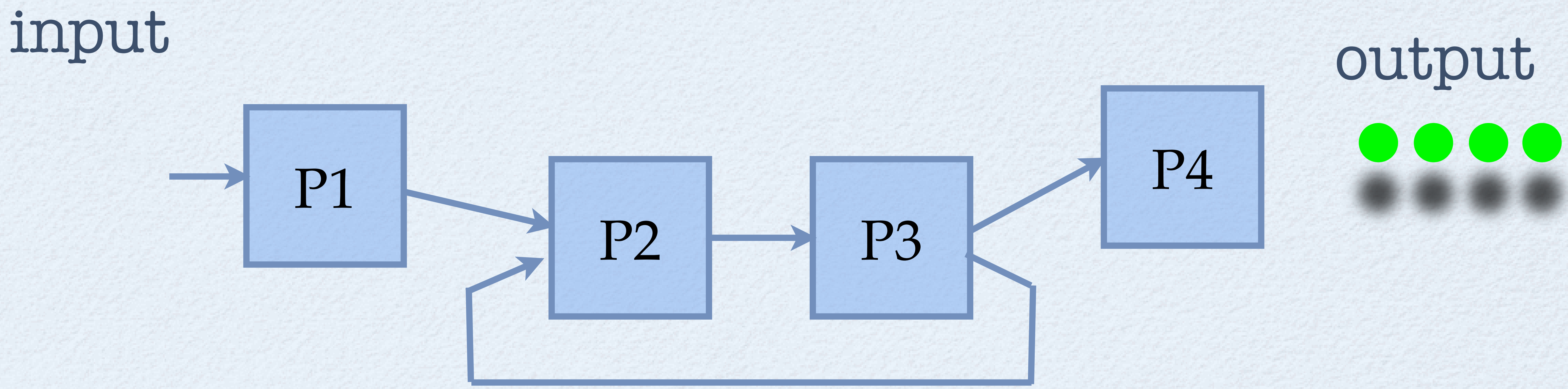
CoreGRID

app = graph of modules

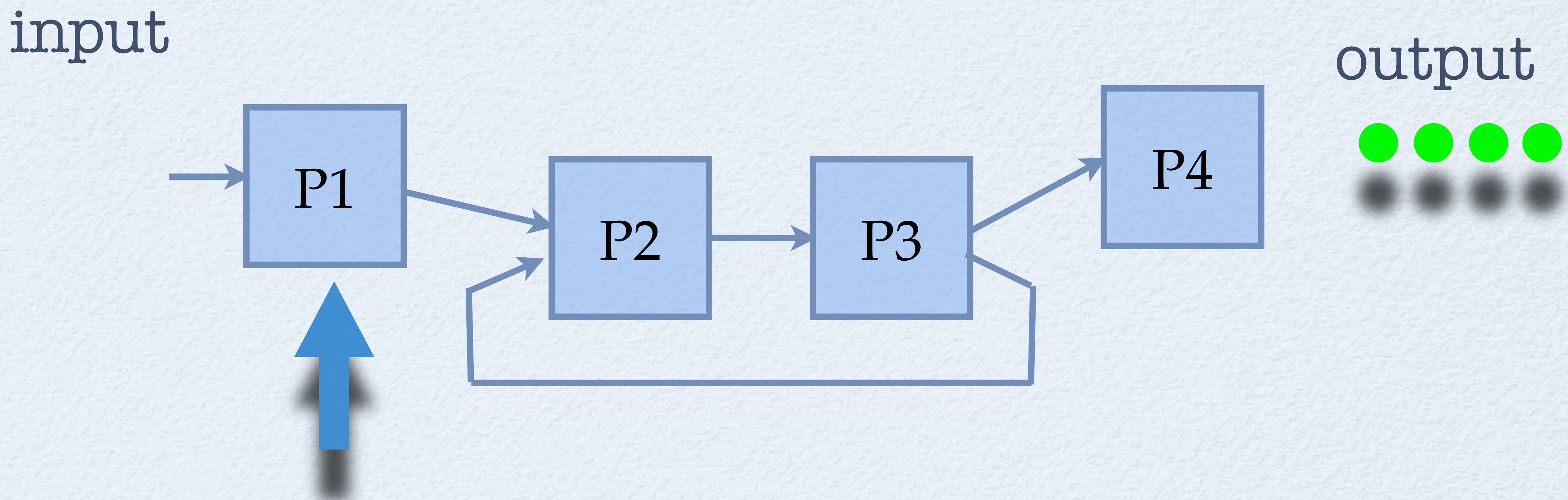


CoreGRID

app = graph of modules



app = graph of modules



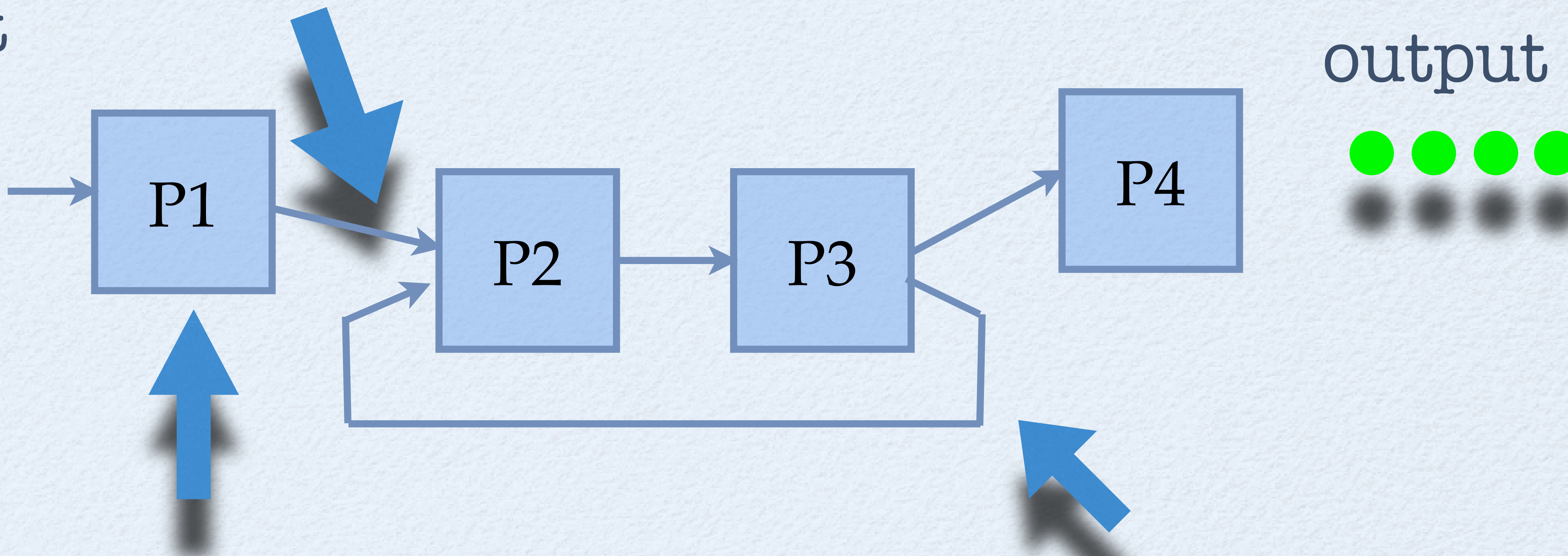
Sequential or parallel module
(native or wrap e.g. MPI, CCM)

app = graph of modules

Programmable, possibly nondeterministic input behavior

input

output

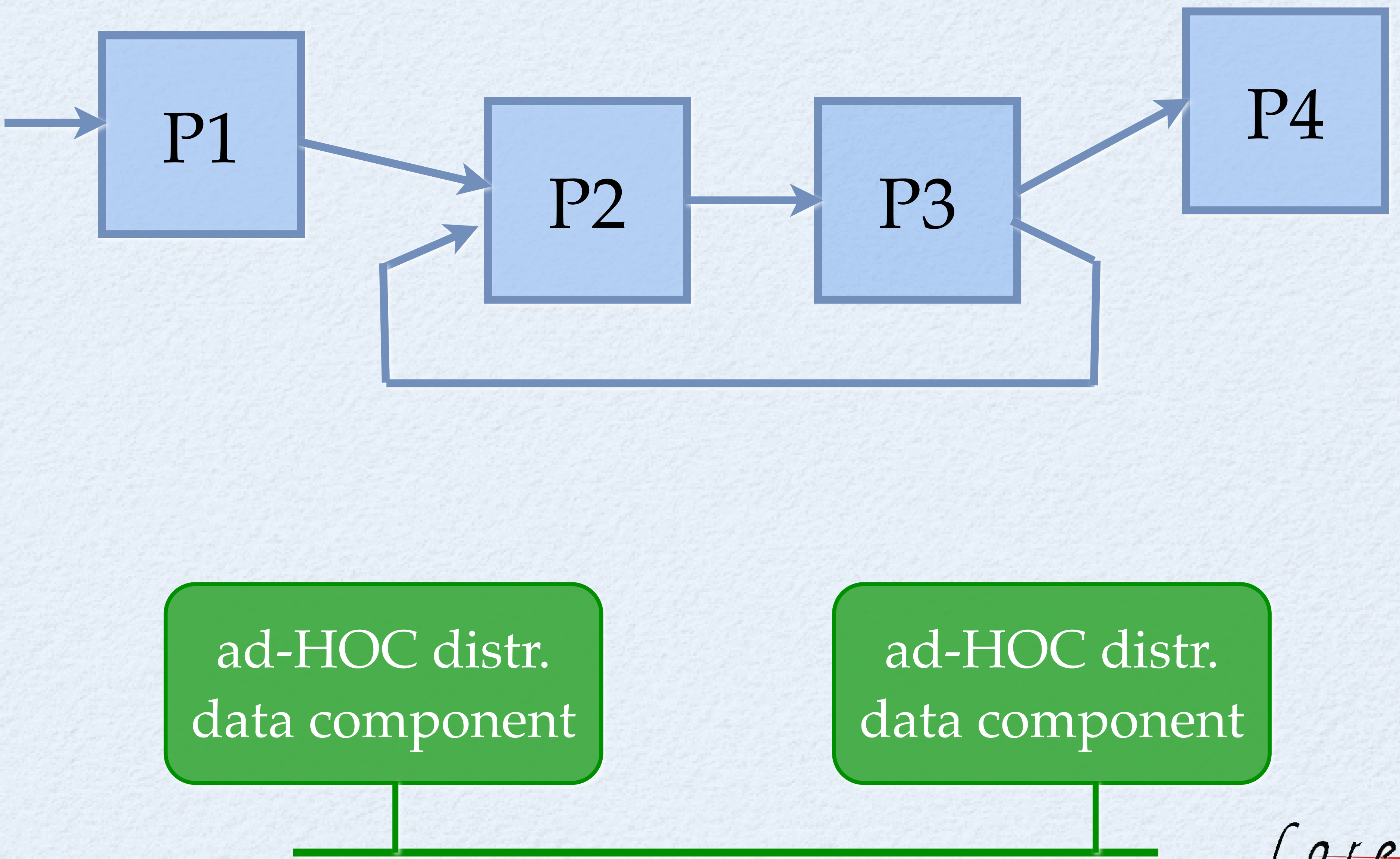


Sequential or parallel module (native or wrap e.g. MPI, CCM)

Typed streams of data items (TCP/IP, Globus, IIOP CORBA, HTTP/SOAP)

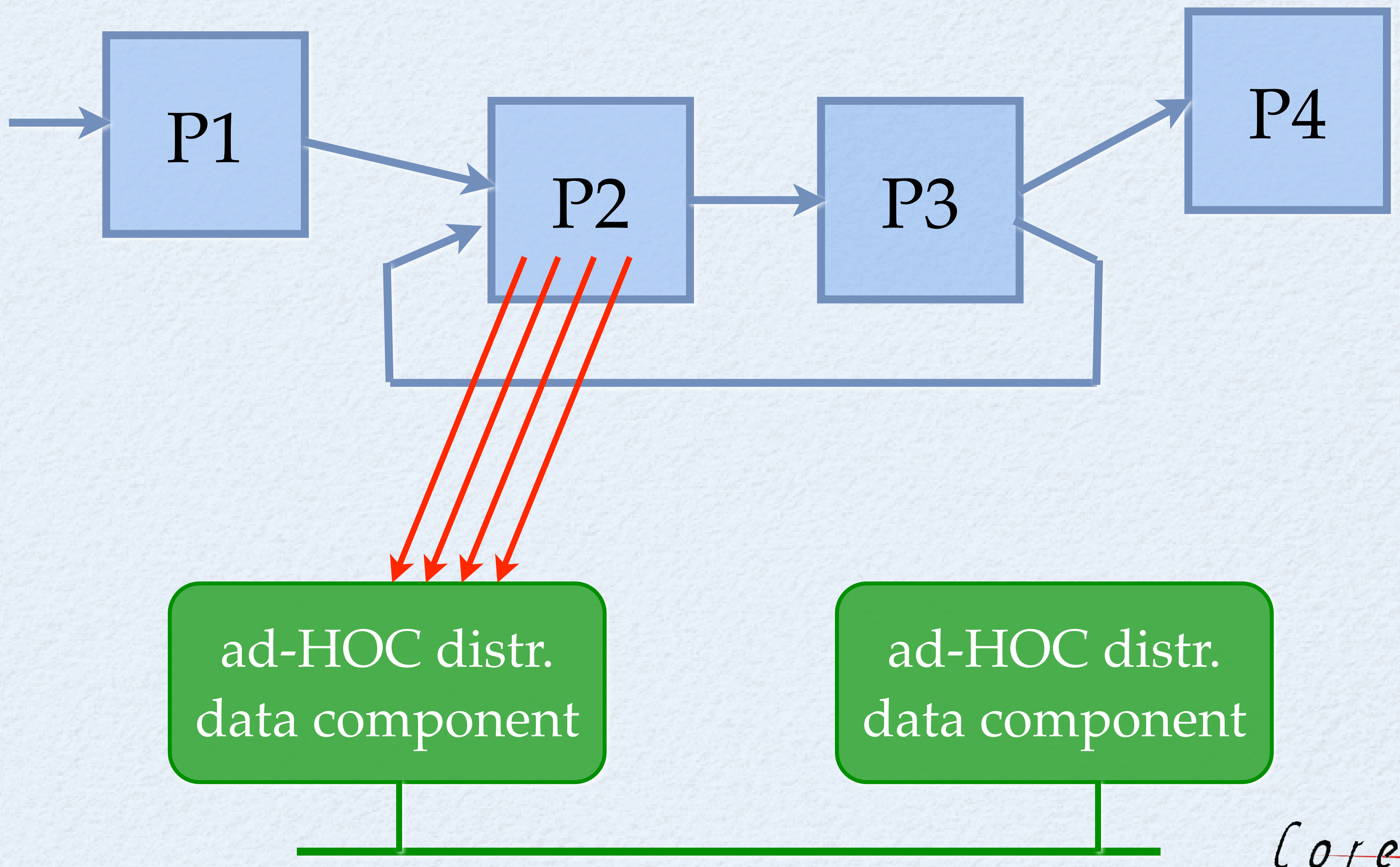
CoreGRID

Supports data sharing



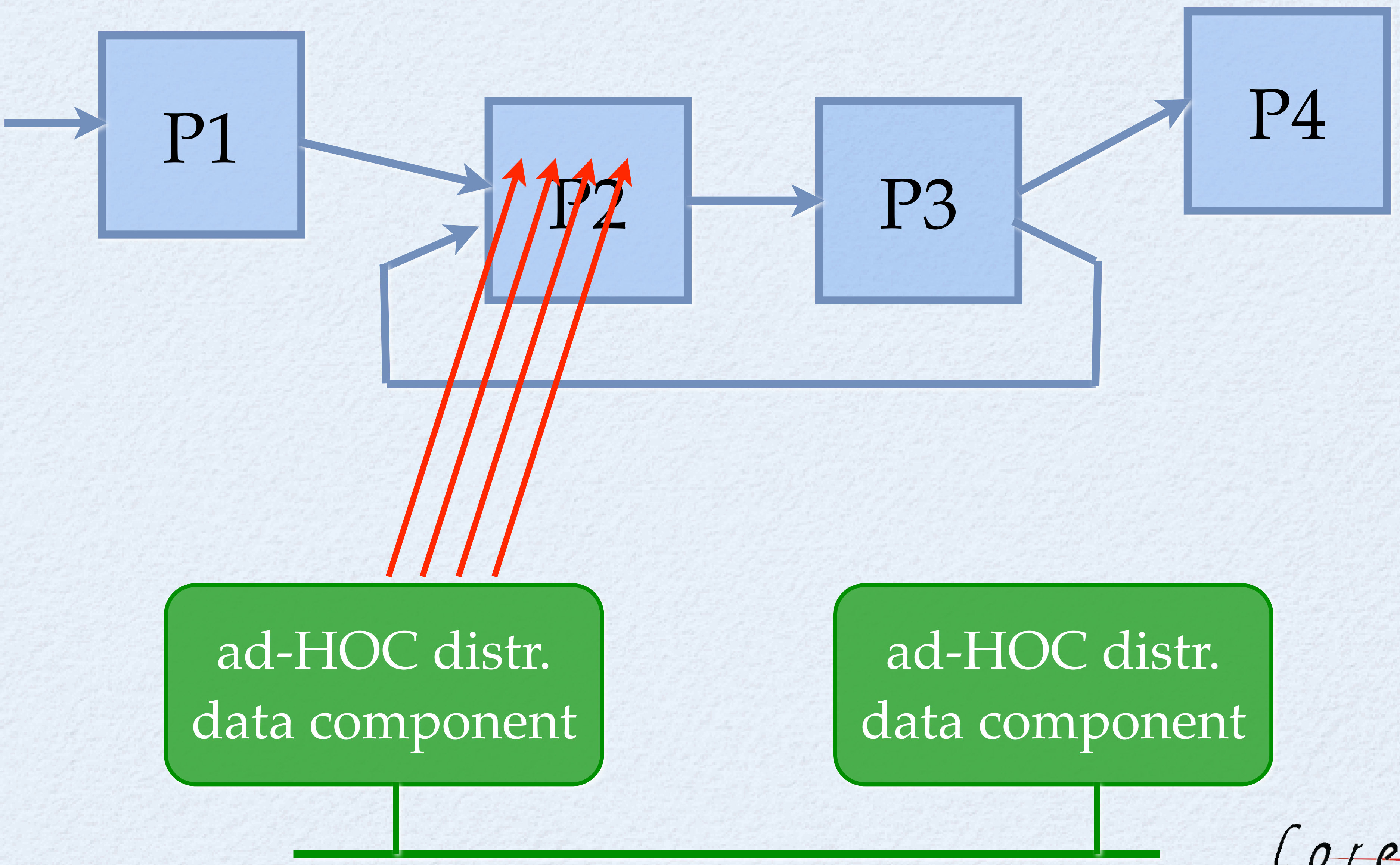
Supports data sharing

1. Shared state within a parmod (attributes)



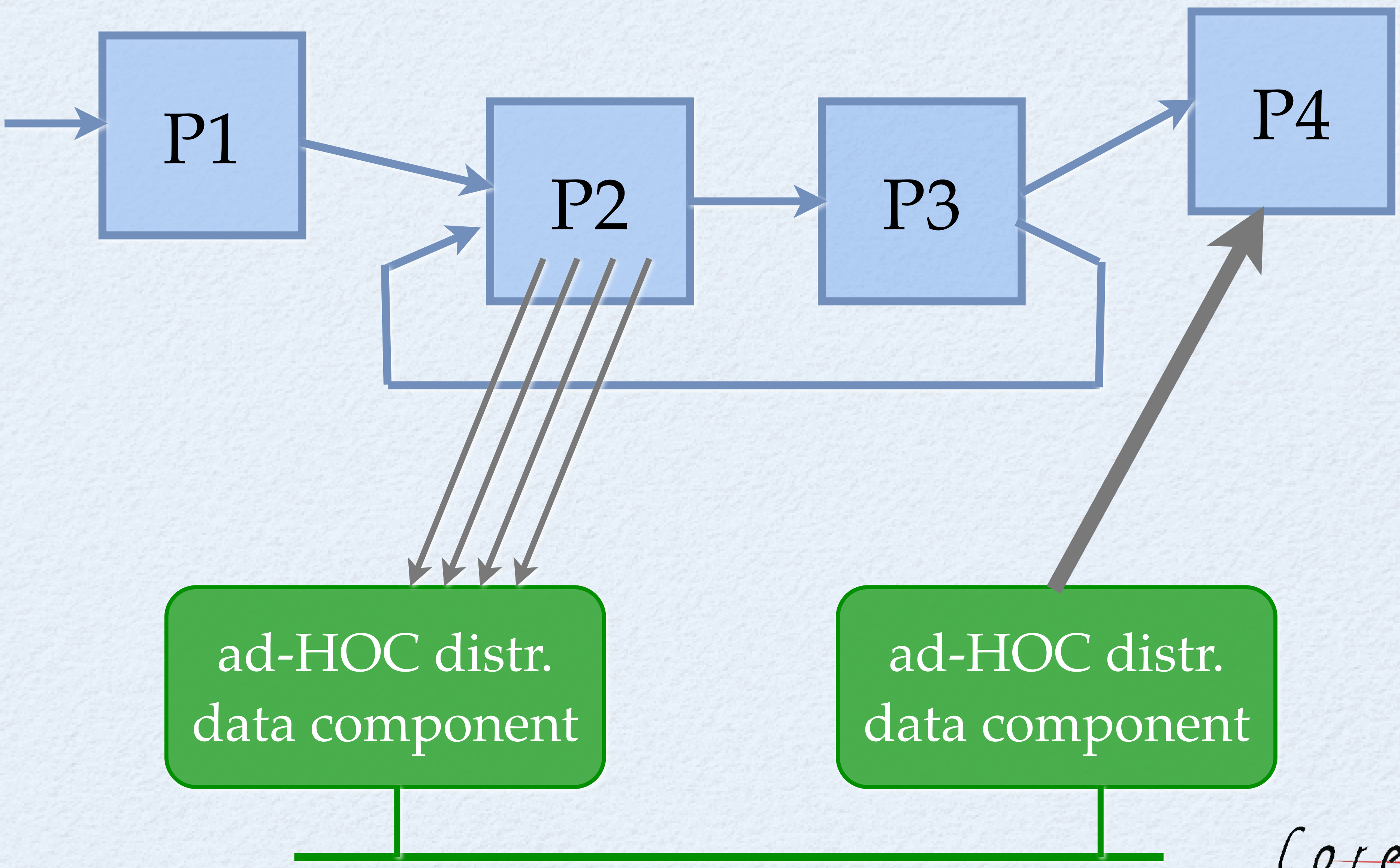
Supports data sharing

1. Shared state within a parmod (attributes)



Supports data sharing

1. Shared state within a parmod (attributes)
2. Shared state among parmods (references)



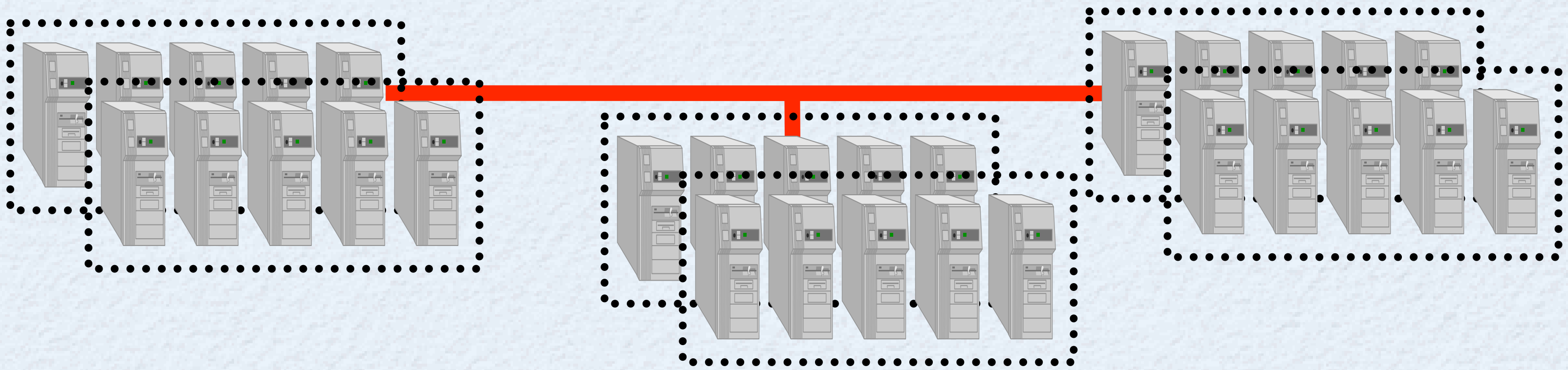
CoreGRID

JuxMem @ INRIA Rennes

- Grid data service
  - P2P JXTA-based prototype
  - Transparent access to data blocks
  - Persistent storage
  - Mutable data: consistency guarantees
  - Active support for peer volatility
- API
  - alloc, map, get, put, lock, unlock

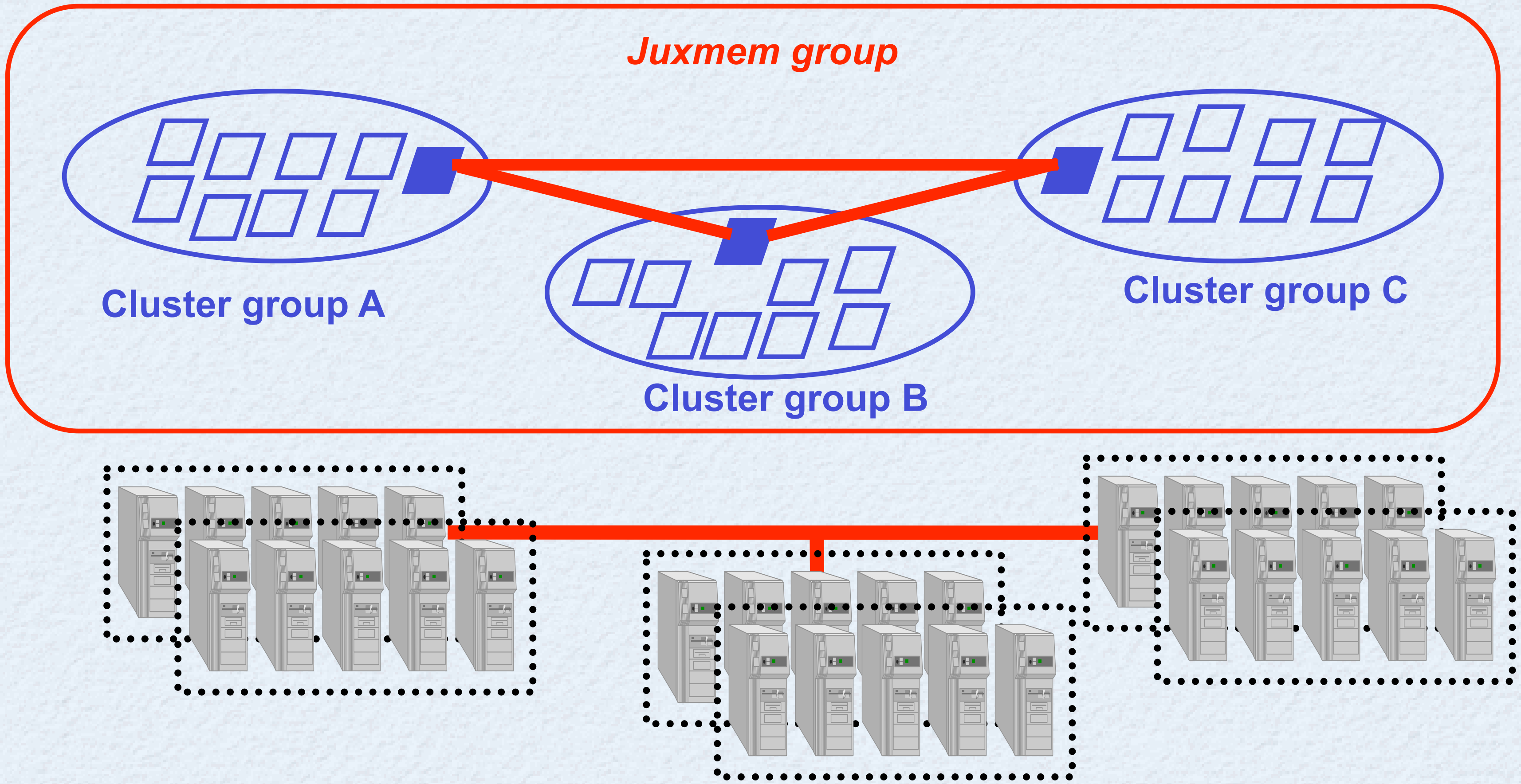
Overview of JuxMem's architecture

- User should define:
    1. on how many cluster replicate data;
    2. how many providers in each cluster,
    3. consistency protocol



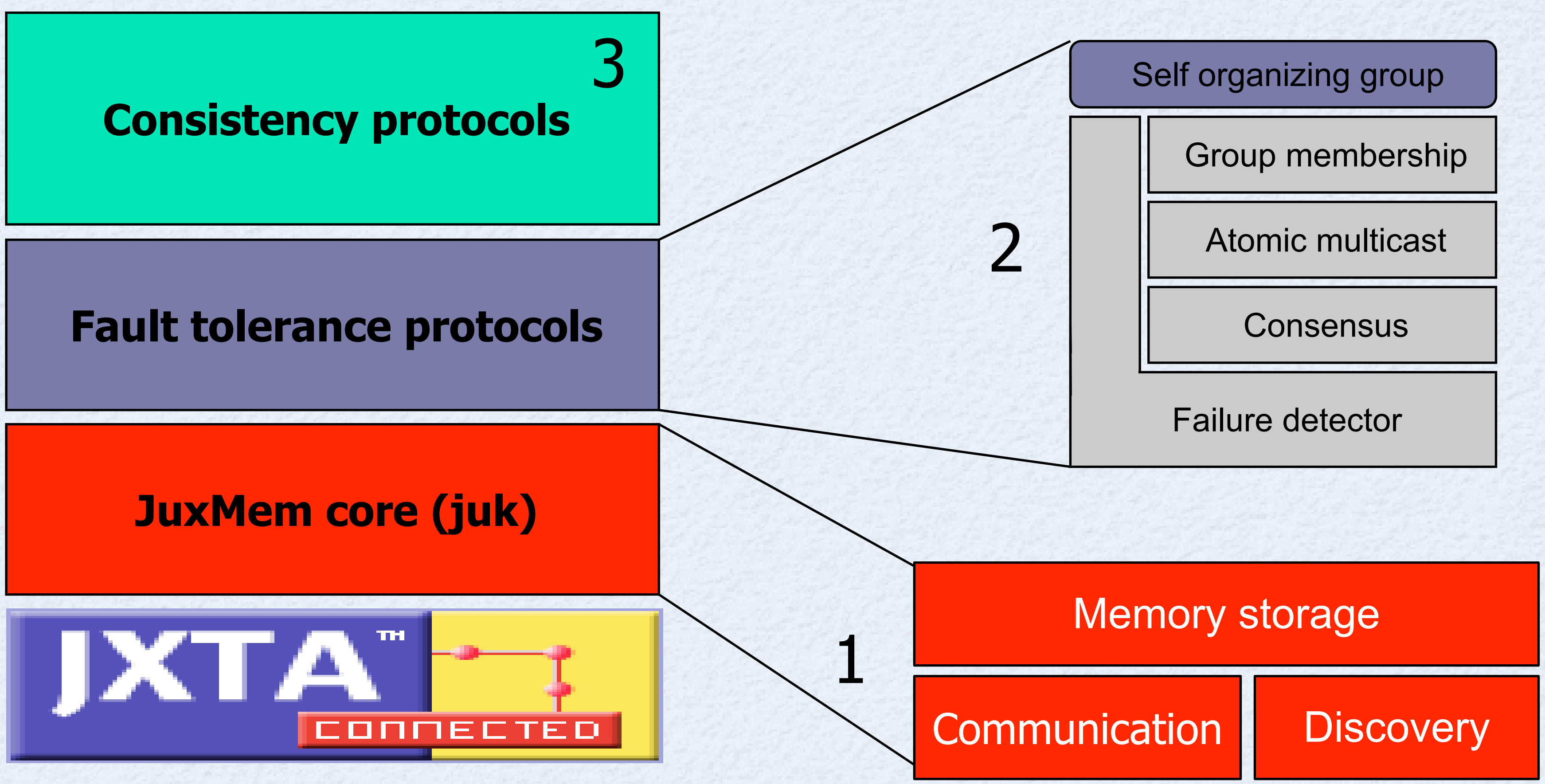
Overview of JuxMem's architecture

- User should define:
  1. on how many cluster replicate data;
  2. how many providers in each cluster,
  3. consistency protocol



JuxMem layers

JuxMem core layers used to test P2P techniques over grid infrastructures



Comparison at hand

No free lunches in nature ...	Cluster sharing (ad-HOC)	Grid sharing (JuxMem)
Throughput	High	High-Medium
Latency	Low	High
Parallel access to a single data item	Read/Write	Read only
Data consistency	No	Yes
Fault-tolerance (data replication)	No	Yes
Dynamically reconfigurable	Yes	Yes
Data location transparency	Yes	Yes

A memory hierarchy for the grid

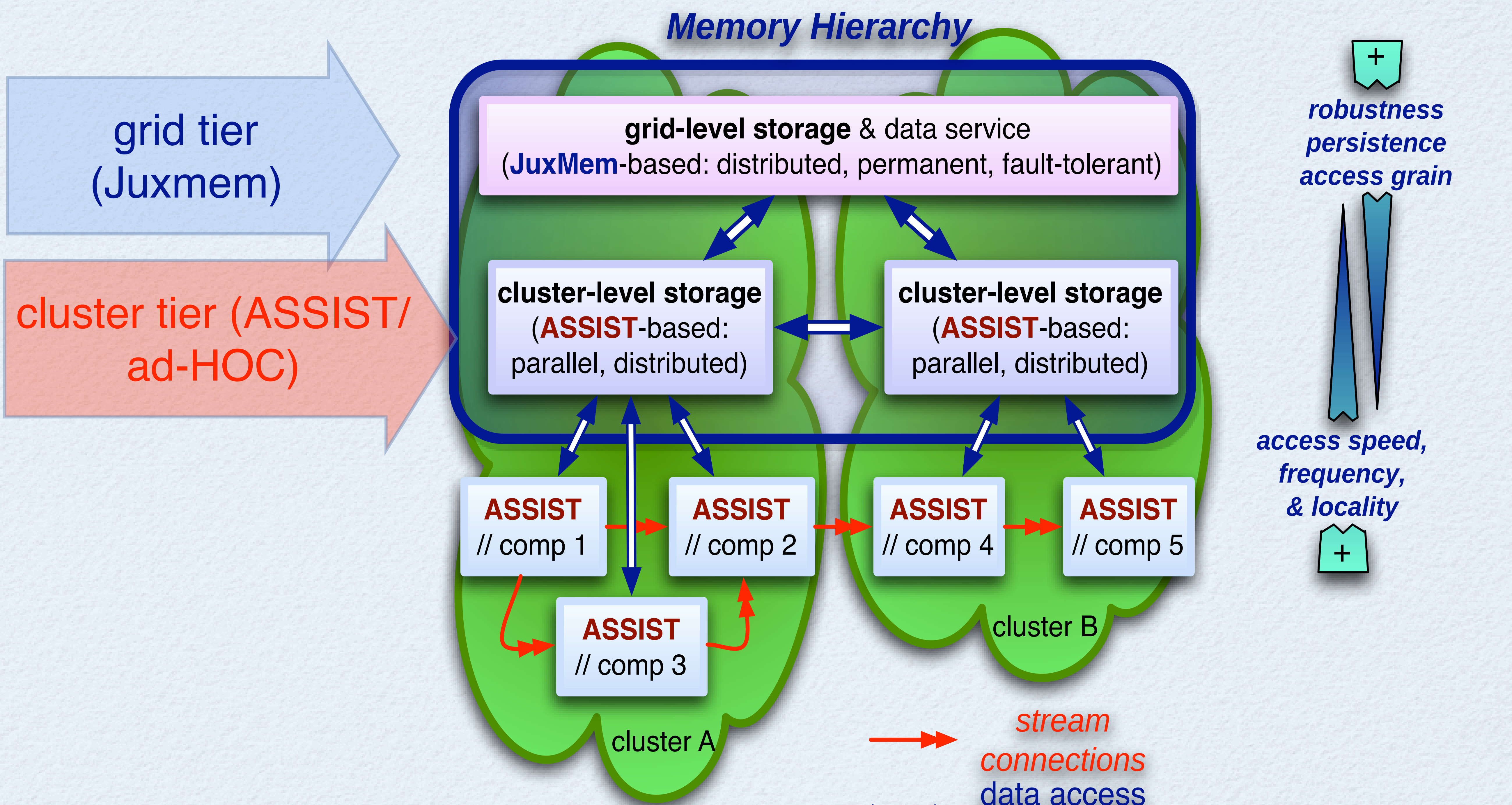
JuxMem and ad-HOC can be organized in a two-tier memory hierarchy

- robust *almost* as JuxMem
- fast *almost* as ad-HOC

provided that

- data locality is promoted by the programming model
- data is transparently exchanged between the two tiers

Integrated architecture



Data locality

- Two kinds
  - classical spatial/temporal locality
  - clustered locality
- Enforced by programming model
  - skeletons/paradigms lead to regular interaction patterns
  - modules/components helps to enforce locality delimitating activities with frequent interactions
    - Fractal, GCM & hierarchic models (provided a proper mapping exists)

Useful for what?

- Sharing across multiple clusters
  - sharing among different applications (persistency)
  - relax co-allocation constraints via stream buffering
    - Direct Acyclic Graphs does not need strict co-allocation
  - data is stored in safe w.r.t. node faults
- Fault-tolerant data storage / checkpointing
  - checkpointing driven by app semantics
  - ASSIST already instruments apps with reconf-safe points for adaptivity (data on ad-HOC is "coherent")

CoreGRID

Prototype & experiments

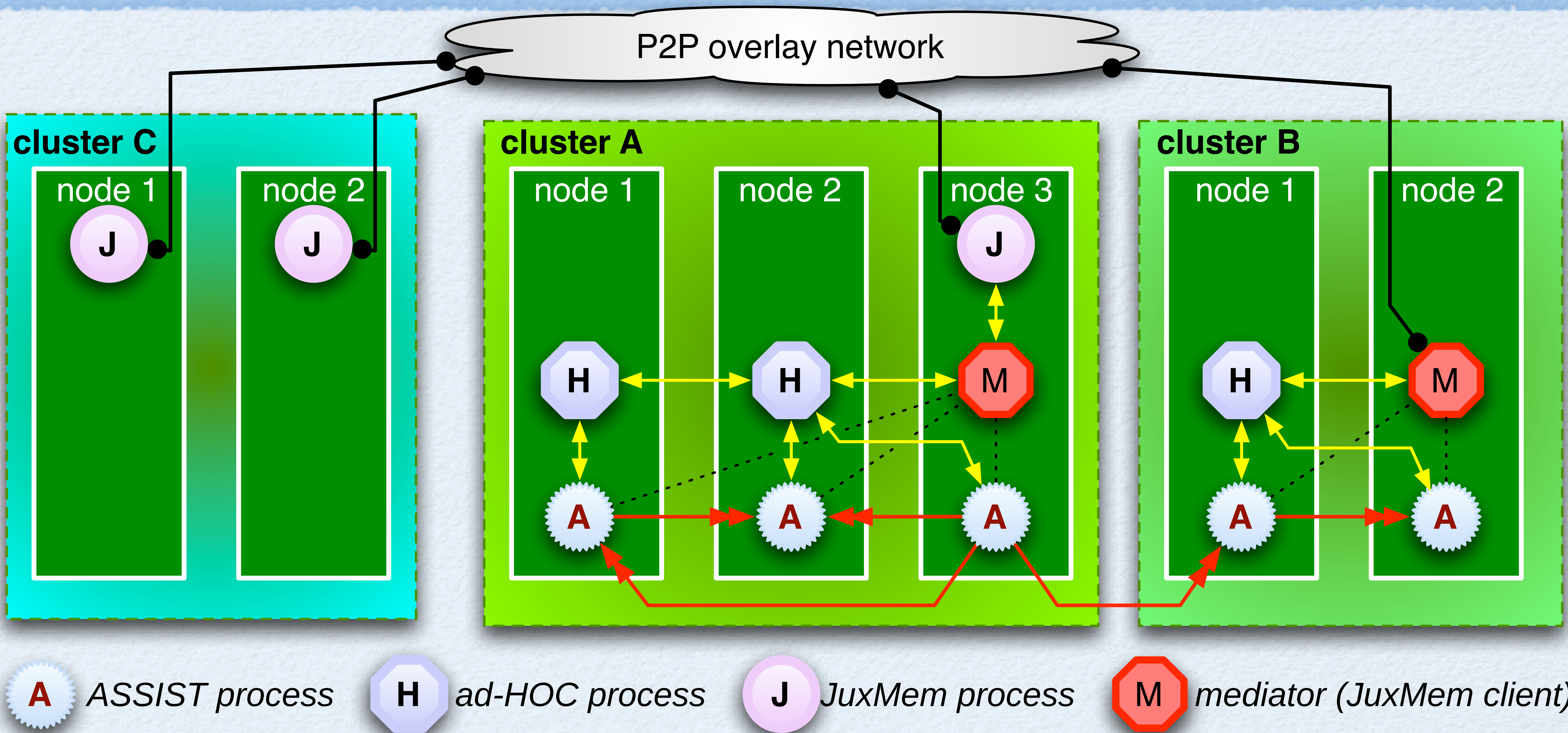
- A preliminary prototype exists
  - developed by students (master thesis)
    - software engineering time not fully predictable
    - CoreGRID does not pay SW engineers
- Experiments are also preliminary
  - focused on the correctness of the system
  - focused on behavior of parts “in insulation”
    - some examples follows

CoreGRID

Prototype & experiments

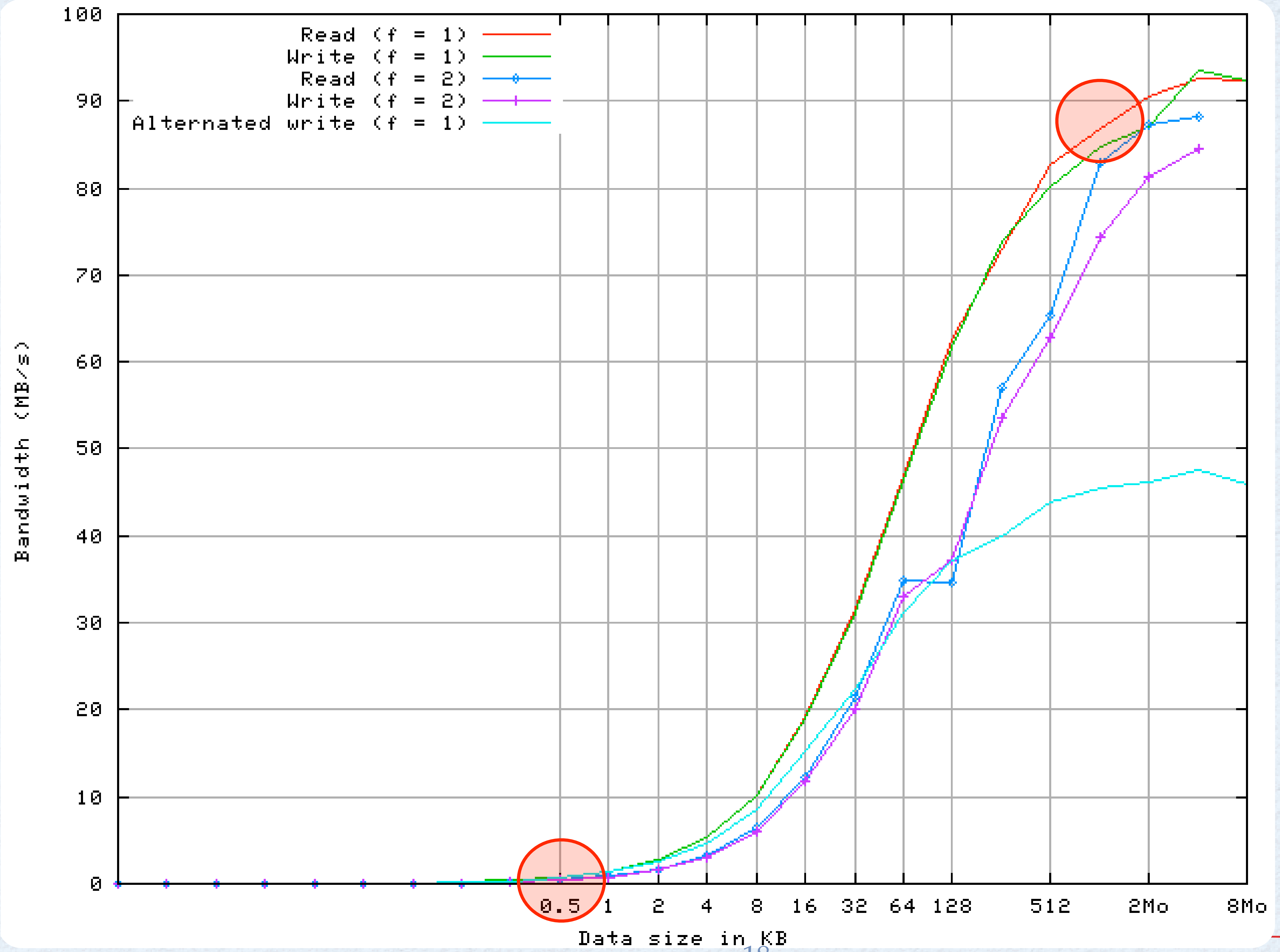
- A preliminary prototype exists
  - developed by students (master thesis)
    - software engineering time not fully predictable
    - CoreGRID does not pay SW engineers
- Experiments are also preliminary
  - focused on the correctness of the system
  - focused on behavior of parts “in insulation”
    - some examples follows

Implementation

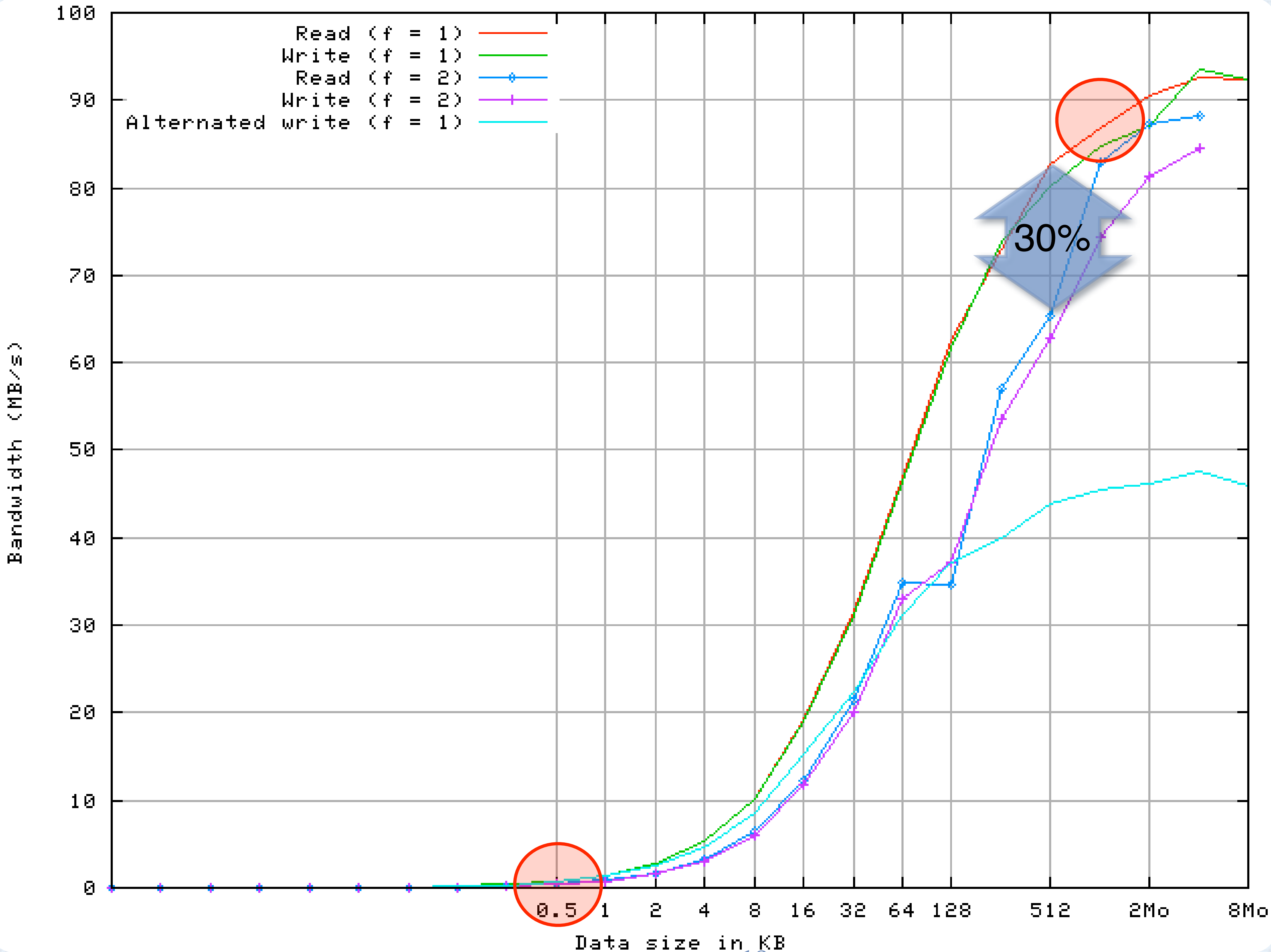


- mediators are triggered by application processes (compiler instrumented)
- they read/write data between the two memory tiers

JuxMem bandwidth (G-Eth)

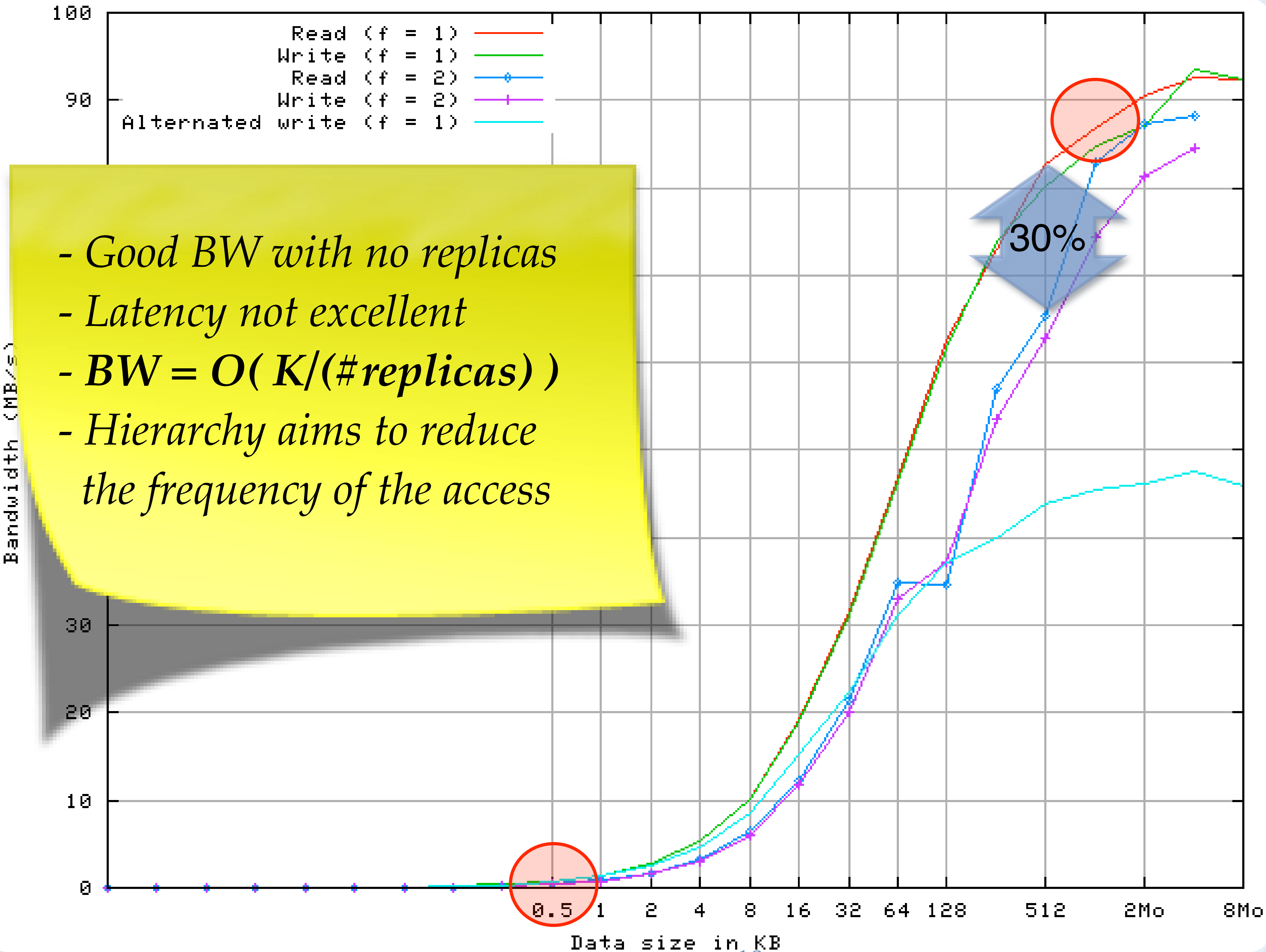


JuxMem bandwidth (G-Eth)



JuxMem bandwidth (G-Eth)

- *Good BW with no replicas*
- *Latency not excellent*
- ***BW = O(K/(#replicas))***
- *Hierarchy aims to reduce the frequency of the access*



ad-HOC figures

Arch/Net/OS	concurrent connections	Msg size (Bytes)	Replies/Sec	net throughput (Bytes/Sec)	net throughput w.r.t. ideal
P4@2GHz Mem 512MB GigaEth Linux ker. 2.4.22	2048	1 M	91	91 M	96%
	3072	512	20 M	10 M	11%
P3@800MHz Mem 1GB FastEth Linux ker. 2.4.18	1024	8 K	1429	11.2 M	90%
	1024	16 K	718	11.2 M	90%

ad-HOC figures

+ 7%

Arch/Net/OS	concurrent connections	Msg size (Bytes)	Replies/Sec	net throughput (Bytes/Sec)	net throughput w.r.t. ideal
P4@2GHz Mem 512MB GigaEth Linux ker. 2.4.22	2048	1 M	91	91 M	96%
	3072	512	20 M	10 M	11%
P3@800MHz Mem 1GB FastEth Linux ker. 2.4.18	1024	8 K	1429	11.2 M	90%
	1024	16 K	718	11.2 M	90%

ad-HOC figures

+ 7%

Arch/Net/OS	concurrent connections	Msg size (Bytes)	Replies/Sec	net throughput (Bytes/Sec)	net throughput w.r.t. ideal
P4@2GHz Mem 512MB GigaEth Linux ker. 2.4.22	2048	1 M	91	91 M	96%
	3072	512	20 M	10 M	11%
P3@800MHz Mem 1GB FastEth Linux ker. 2.4.18	1024	8 K	1429	11.2 M	90%
	1024	16 K	718	11.2 M	90%

+ ~1000%

CoreGRID

ad-HOC figures

- *Good BW for small data size*
- *Good latency (not shown)*
- *Good support for concurrent connection (firewalls & co)*
- ***No data replication, no FT, cluster-oriented***

		Msg size (Bytes)	Replies/Sec	net throughput (Bytes/Sec)	net throughput w.r.t. ideal
		1 M	91	91 M	96%
GigaEth Linux ker. 2.4.22	3072	512	20 M	10 M	11%
P3@800MHz Mem 1GB FastEth Linux ker. 2.4.18	1024	8 K	1429	11.2 M	90%
	1024	16 K	718	11.2 M	90%

+ 7%

+ ~1000%

Conclusions

- Both JuxMem (F) and ASSIST/ad-HOC (I) implement data storage services.
  - They exhibit a similar API but complementary aims and features.
- They can be composed to set up a parallel, distributed, efficient, fault-tolerant memory hierarchy
  - **Real integration** of existing (and complex) software developed by different CoreGRID partners
  - It enables the experimentation of architectural solution for high-performance robust data services for the grid
  - It can be used as robust storage for checkpoints
- Aims to understanding how memory hierarchies work in grid env., and how they are related to programming model (beyond ASSIST ...)