Coherence of Shared Replicated Data within Groups of Remote Cooperating Processes

Georges Brun-Cottan

Projet SOR – INRIA Rocquencourt (University Paris 6)

Now at

Michtom School of Computer Science, Brandeis (Liuba Shrira)

PhD advisor: Mesaac Makpangou

University advisor : Claude Girault

SOR group leader : Marc Shapiro

`http://www.cs.brandeis.edu/~bruncott/talks/brandeis-10-28-1998.ps`

outline

Targetted Applications – Hypothesis

- ***Cooperative Application above Internet***
- **Collaborative Work, Interactive Remote Learning, Virtual World, Multi-users games.**

Hypothesis:

- **Cooperation** $\Rrightarrow$
  - Sharing the same information.
  - Application progression rely on cooperation.
- **Shared information: determinist specification (ADT, object).**
- **System: fixed processus, latency is high and non predictable.**
- **Application :** ***self-made fault-tolerance if required/possible.***

Replication

☞ Hypothesis ⇒ local copies (total replication.)

Which consistency ?

- **¬ sequential consistency (USENET, email, Internet games)**
  ⇒ no more transparency
- **Weak-consistency :** *epsilon*-**serialisibility, eventual consistency, time/space-bounded, entry-consistency**
  ⇒ *ad hoc* solution
  - no more transparency
  - depend on the application code
    ⇒ ***Complexity !*** . . . ***+ non predictible behavior !***

Weak consistencies? Looking at the state of the art

- **Consistency is a correctness criterion.**
  - Asynchronism + determinism $\Rightarrow$ set of constraints (predicates) on the order the operations are executed (consistency model).
- **A consistency model is an attribute of the function managing the sharing of the information (memory, object, network).**
- **Weakness of such a function means being more or less conservative on the kind of schedule this function is willing to "accept/generate".**

consistency issues: state of the art

- **Properties :**
  - *epsilon*-serialisibility, causality.
- **Topology (DBMS, atomic objects, DSM) :**
  - (DBMS, atomic objects) : fine-grain CC, predicate locks, granular locks.
  - (DSM) : diff (*Munin*), replicated monitor (*entry-consistency*), hybrid protocol (page/objet).
- **Semantics (DBMS, atomic objects, DSM, CATOCS):**
  - isolation degree, labeling, algebric properties (commutativity), dependance (serial specification), metrics.

☞ **consistency = replication (perception) + CC, semantics (operation + application).**

Goal

Adapt and *Reuse*
the consistency managment.

Contribution

- **A model of sharing : cooperative replication**
  - Replicas are autonomous, but still controlled.
- **An execution model based on replicated history.**
  - Model for weak consistency.
  - Expose the important optimisations.
- **Few system principles $\Rightarrow$ one "system" component (consistency manager + Core architecture)**
  - implements the full cooperation protocol
  - provides a deterministic contract
  - is bound at execution,
  - is reusable

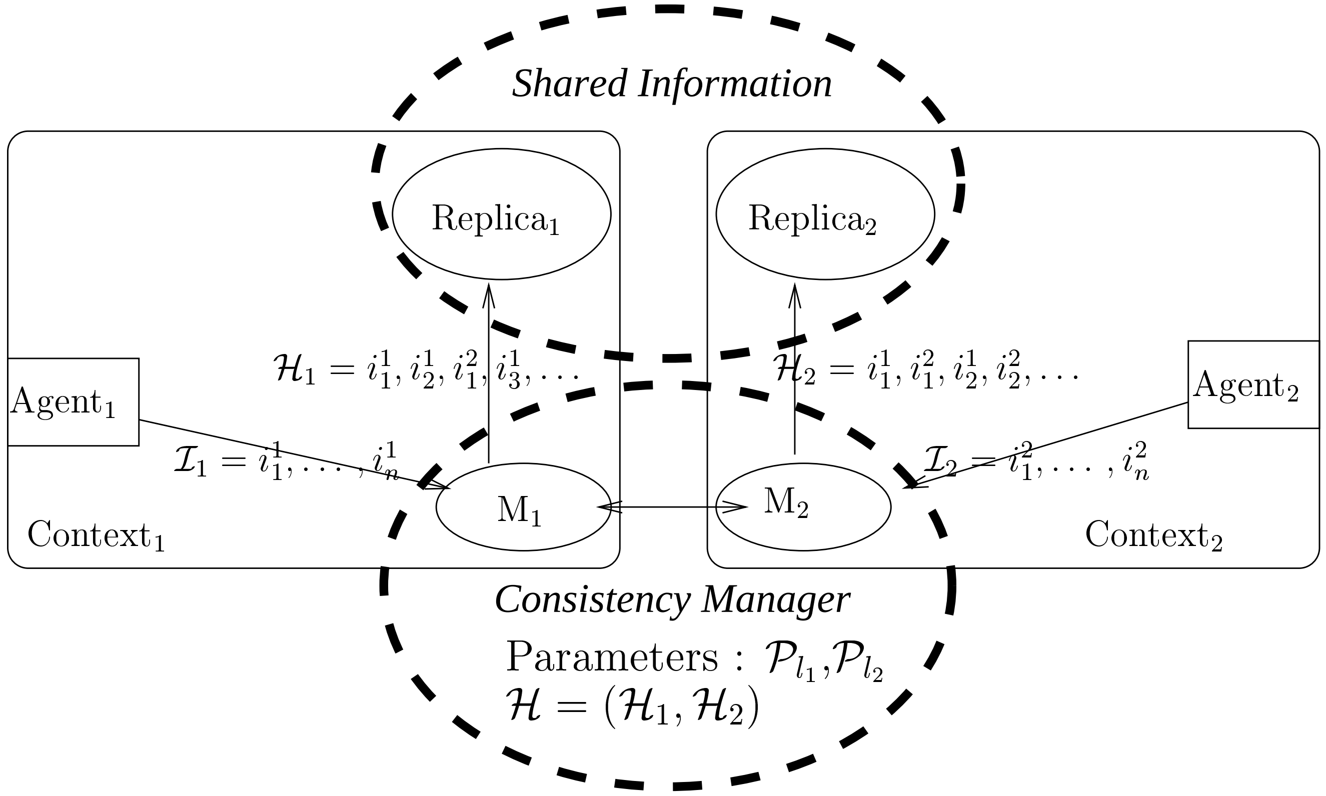
Outline

Core Architecture : rational

Adapt and Reuse

- **we decompose the consistency criterion and isolate the global criterion.**
- **we factorize the correspondant criterion implementations and encapsulate the global criterion (distributed algorithmics) in one single component.**
- **we design and build system abstraction needed to support dynamic binding and reuse.**

Core principles



Decomposition of the consistency criterion

Core criterion : $\mathcal{P}^i_{l_1}(\tilde{\mathcal{H}}_i \bullet \mathrm{op}) \wedge \left[\mathcal{P}^i_{l_2}(\tilde{\mathcal{H}}_i \bullet \mathrm{op}) \vee \mathcal{P}_g(\tilde{\mathcal{H}}, \tilde{\mathcal{H}}_i \bullet \mathrm{op})\right]$

☞ **3 predicates :**

1. $\mathcal{P}_{l_1}$ (binary) is true when locally, the data respects its serial specification (local correctness).
2. $\mathcal{P}_{l_2}$ (binary) is true when local correctness is enough to achieve global correctness.
3. $\mathcal{P}_g$ is true whenever $\mathcal{P}^i_{l_1}(\tilde{\mathcal{H}}_i \bullet \mathrm{op}) \wedge \mathcal{P}_g(\tilde{\mathcal{H}}, \tilde{\mathcal{H}}_i \bullet \mathrm{op})$ achieve global correctness.

Example : Replicated semaphore

- **State (replica) : an integer (N).**
- **2 methods : P, V.**

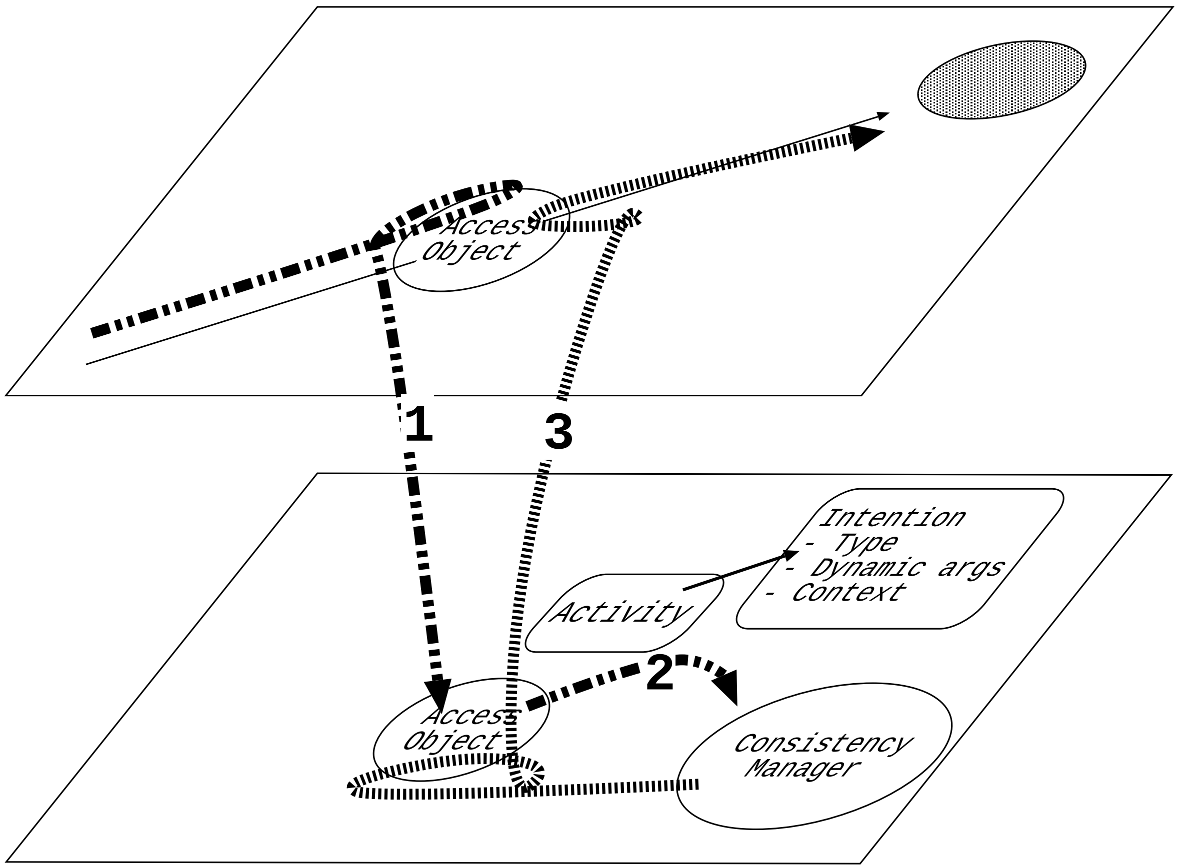
- $\mathcal{P}_{l_1}$ **: true on i if i issued a V operation or if $N_i > 0$.**
- $\mathcal{P}_{l_2}$ **: false for every P operation.**
- $\mathcal{P}_g$ **: true if P operations are totally ordered.**

Factorization (1/2)

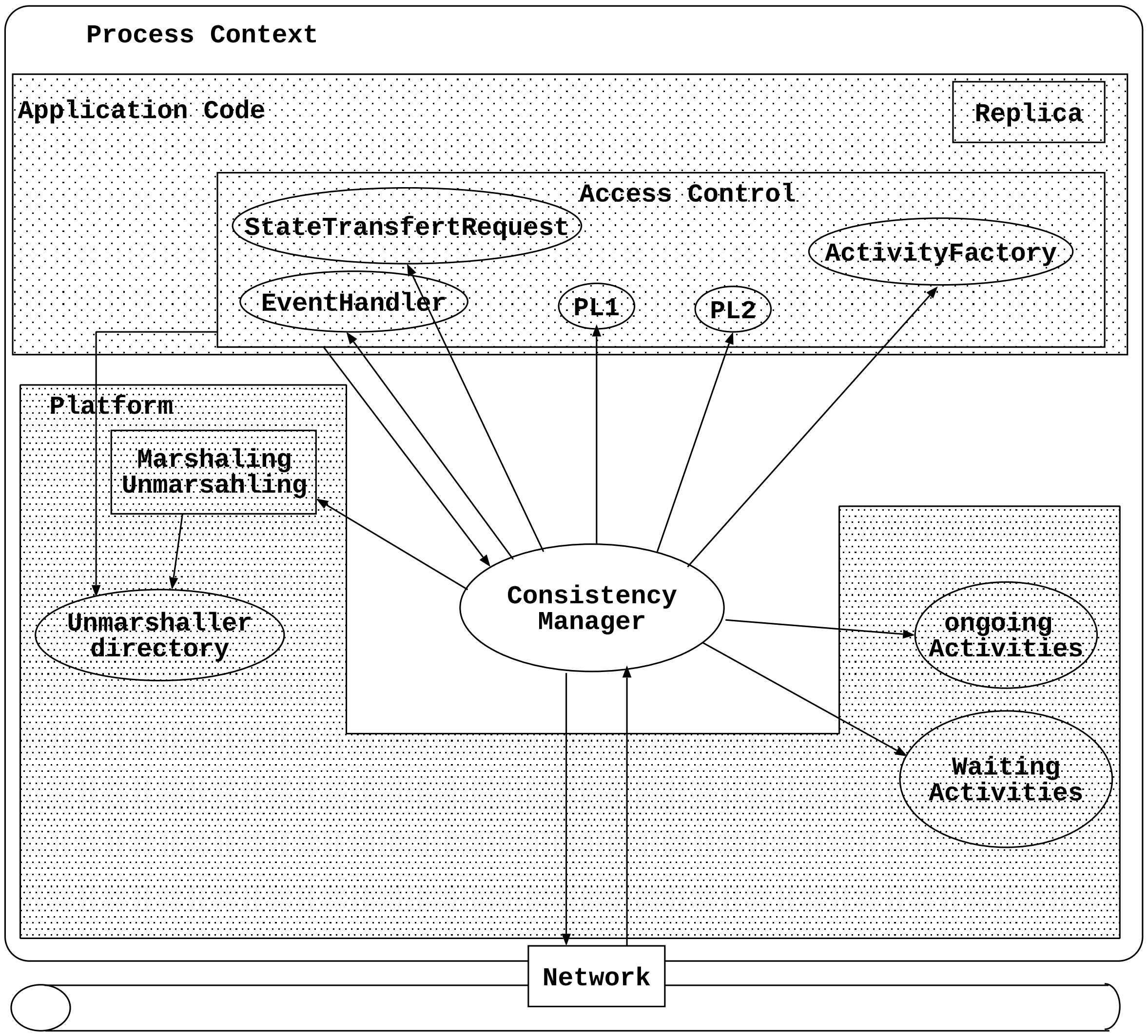
☞ **5 elements :**

- **Consistency manager ($\mathcal{P}_g$) (reusable)**
  - scheduler + group communication.
- **Intention (+/− reusable)**
  - encapsulate the operation semantics.
- **Predicates $\mathcal{P}_{l_1}$ and Predicates $\mathcal{P}_{l_2}$**
  - 2 functions parameterized by the local history (replica state) and one *intention*.
- **Activity (+/− reusable)**
  - encapsulate the context of the operation execution.

Factorization : application to a replicated object (2/2)



Core Architecture

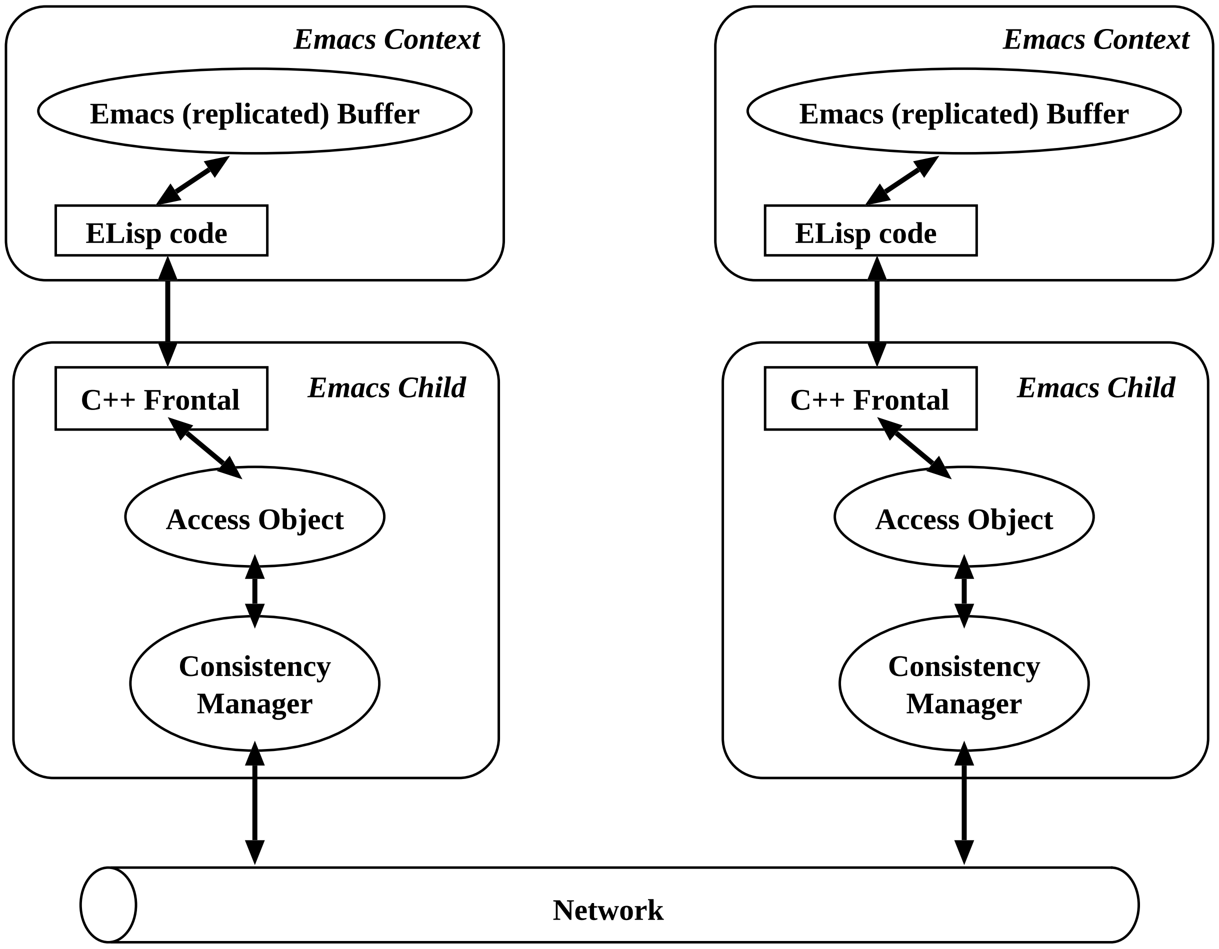

Process Context
Application Code
Replica
Access Control
StateTransfertRequest
ActivityFactory
EventHandler
PL1
PL2
Platform
Marshaling
Unmarsahling
Consistency
Manager
Unmarshaller
directory
ongoing
Activities
Waiting
Activities
Network

Outline

Implementation (C−−)

- **complete platform (name service,(un)marshaling, groups communication, (multi)RPC).**
- **library of consistency manager classes(3).**
- **library of intentions and activities classes.**
- **support for replicated object on top of Core.**
- **toys applications (shared resource, counter, array),**
- **more "serious" application : cooperative editing "on top" of Emacs.**

“Replicated” Emacs



Applying **Core** principles to Emacs

- **State (replica)**
  - an Emacs buffer
  - structured as a tree (LaTeX hierarchy).
- **2 methods : read, write.**

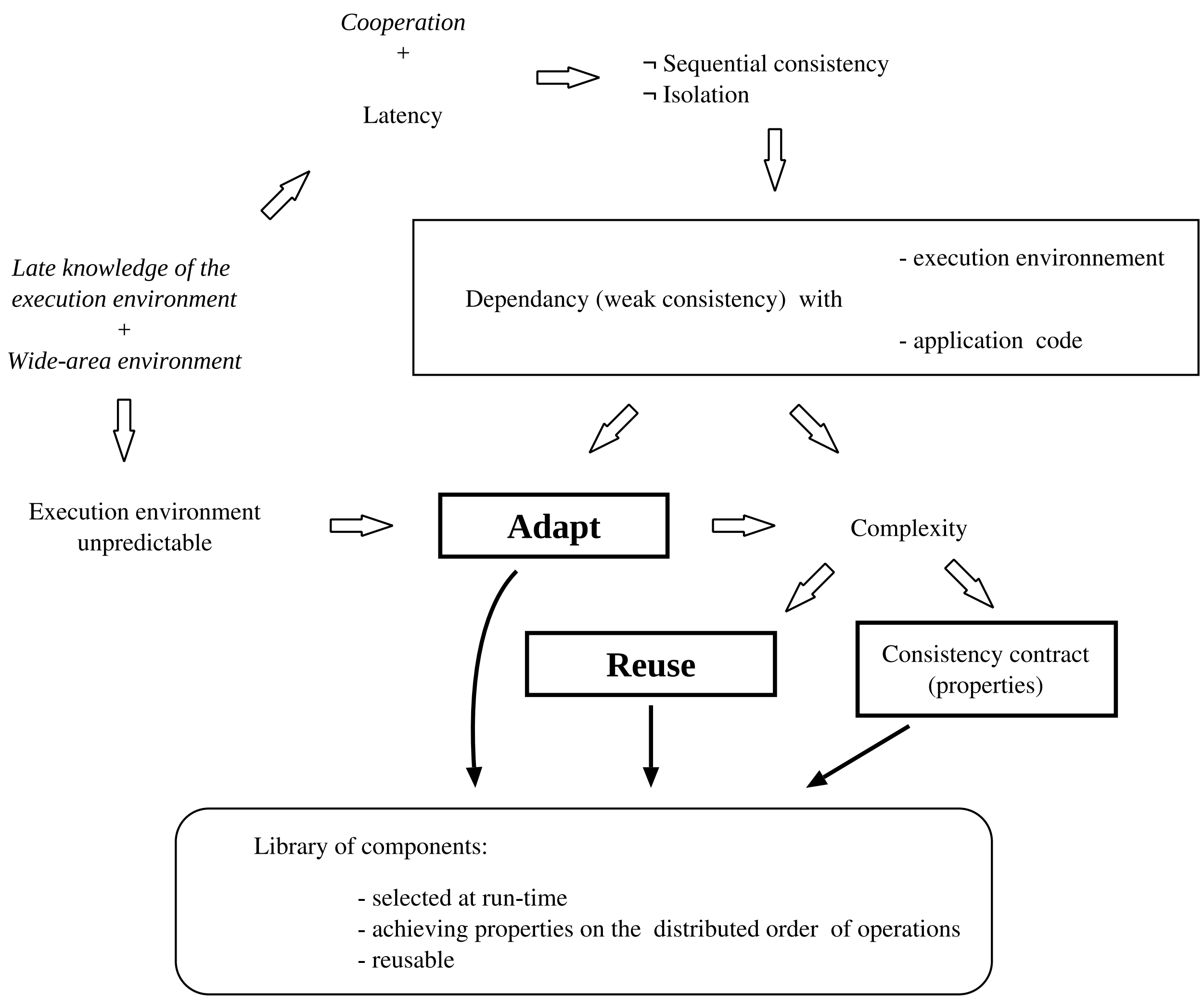
- $\mathcal{P}_{l_1}$ **: true if two overlapping parts are not concurrently accessed in writing.**
- $\mathcal{P}_{l_2}$ **: false for every write operation.**
- $\mathcal{P}_g$ **: true if write operations are totally ordered and if read operations are FIFO ordered (*hybrid consistency*).**

Applying **Core** principles to ACM

- **ACM : Air combat simulation**
- **4 dimensions space (x,y,z,t).**
- **Control of the divergence on the avatar's position (plane, amnos).**
- **Simple metrics: individual position, imprecision on the global virtual time.**
- **Huge latence (¿ 500ms) : good results with a bit of faith :-). Better than with DIS (*best effort*).**

Outline

Intellectual Reasoning



Conclusion

- **Better understanding of consistency in the wide-area cooperation settings.**
- **First step to integrate the consistency management in the system.**
- **Results looks nice even though the platform performance itself is horrible (but all benchs have been done over a LAN (FDDI)).**

Future

- **Benchs in a "real" high latency setting.**
- **Tackle current limitations :**
  - Changing at run-time the consistency manager, composing various managers (clusters), integration (persistence, fault-tolerance), invocation duplication.
- **continuing the studies of virtual world (education, games, enterprise) : soft real time.**
- **putting transaction and optimism in action.**