A Time-Coherent Model for the Steering of Parallel Simulations

Esnard, **M. Dussere**, O. Coulaud

Scalapplix Project Inria
EuroPar 2004, Pisa, Italy.
http://www.labri.fr/epsn

INRIA FUTURS

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
MINISTERE DE LA RECHERCHE

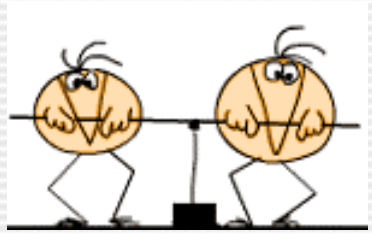
Action Concertée Incitative
[ACI]
Globalisation des Ressources Informatiques et des Données
[GRID]

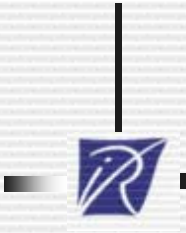
ACI GRID

LaBRI

Outline

1. Introduction
2. Environment for the Steering of Parallel Simulations
3. High-Level Model of Steerable Simulations
4. The Time-Coherence Problem
5. Preliminary Results
6. Conclusion & Prospects

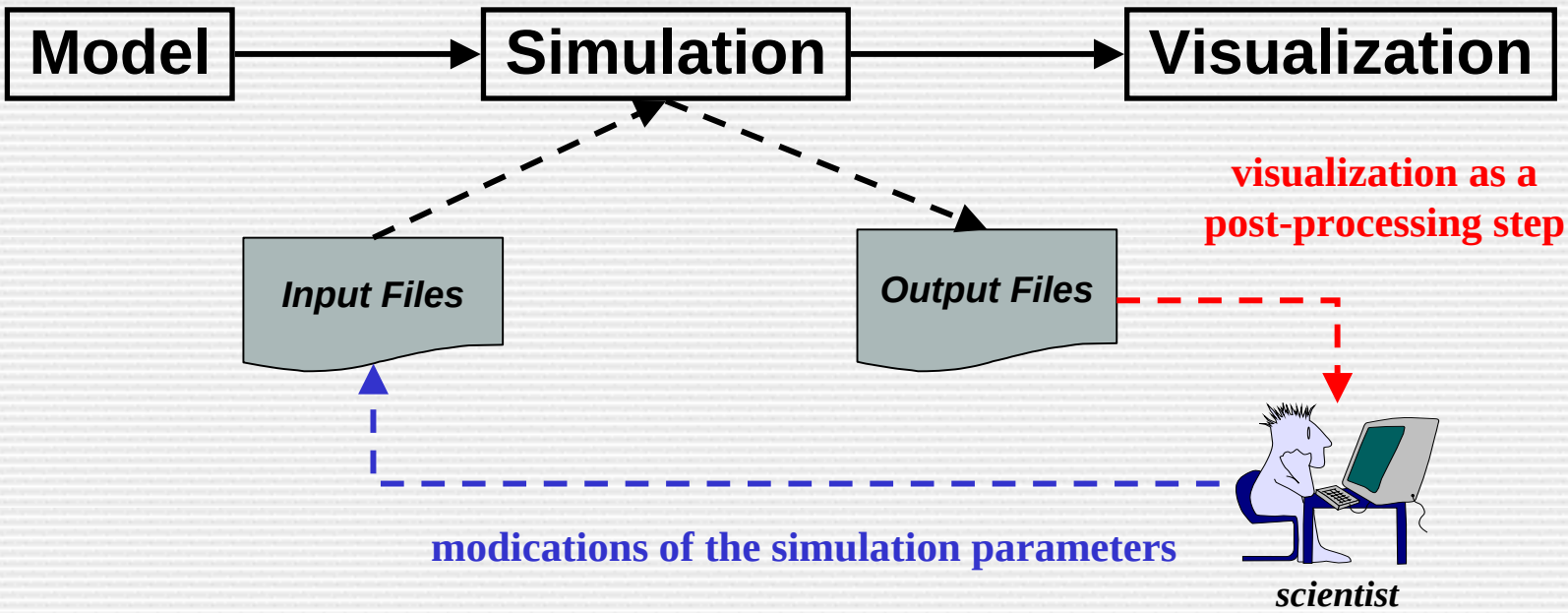




Introduction

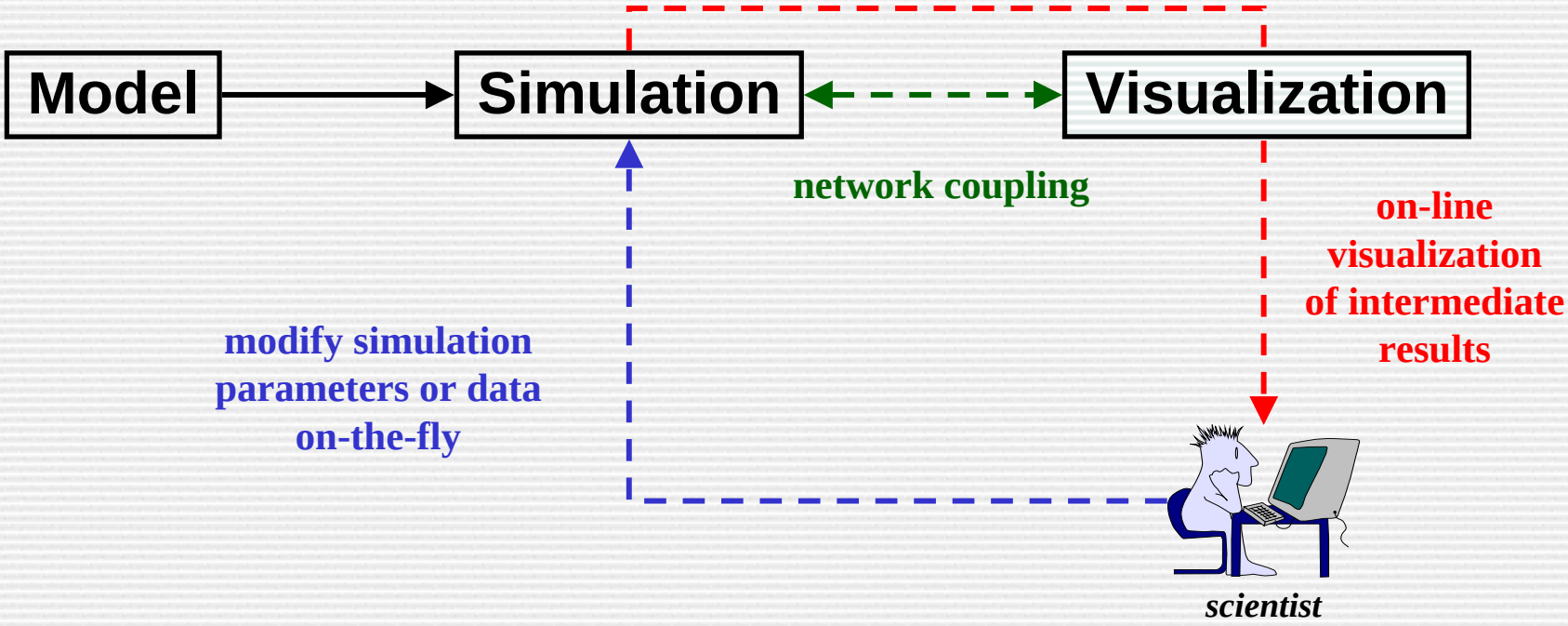
The Batch Approach

- Scientific visualization plays a central role in analysis of data generated by scientific simulations.
- The « batch » sequential work-flow



Computational Steering

- A more interactive approach
- A software environment to couple the simulation with a remote visualization system (through the network)



Context

- Parallel numerical simulations
  - C/C++/Fortran
  - MPI, PVM, *etc.*
  - data generated at each step can be very huge

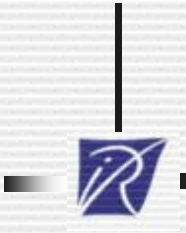
  ⇒ Efficient and time-coherent extraction of data

parallel simulation

- Scientific visualization
  - visualization processing can be very complex and intensive (e.g. isosurface)
  - need of parallelism (cluster vs SGI, reality center, tiled-display wall)

  ⇒ Parallel data redistribution

visualization environment



EPSN

Environment for the Steering of Parallel Simulations

What is EPSN?

A software environment for the steering of numerical simulations.

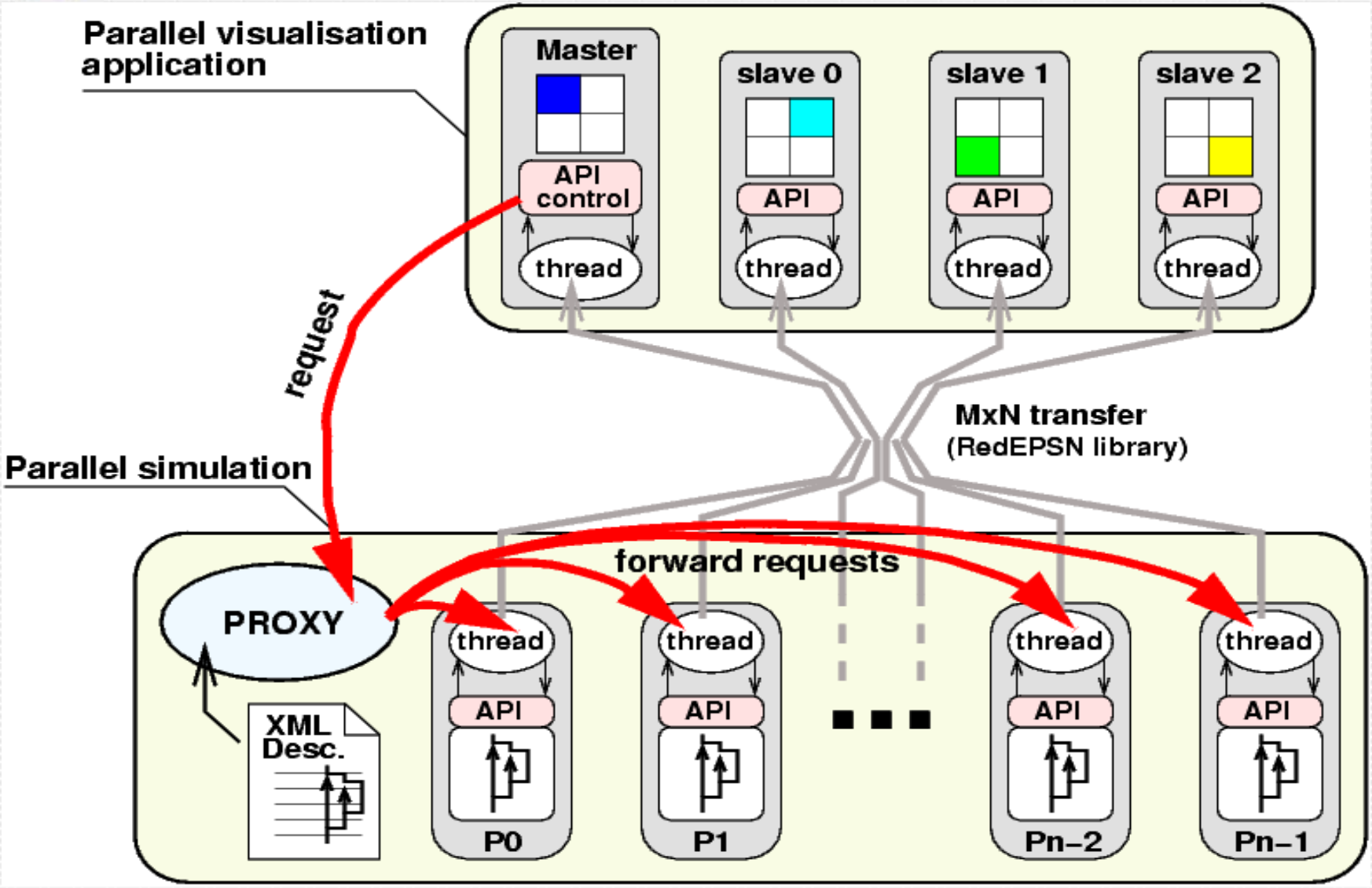
- Parallel simulations
- Parallel visualization systems

- Remote control (play/stop)
- Data collection for the on-line visualization of intermediate results
- Modification of parameters and data on-the-fly
- Invokation of actions in the simulation code

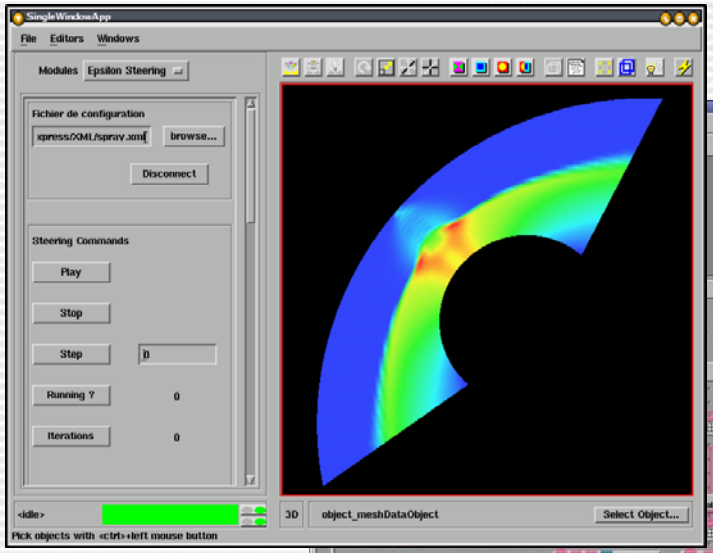
Keypoints of the EPSN Environment

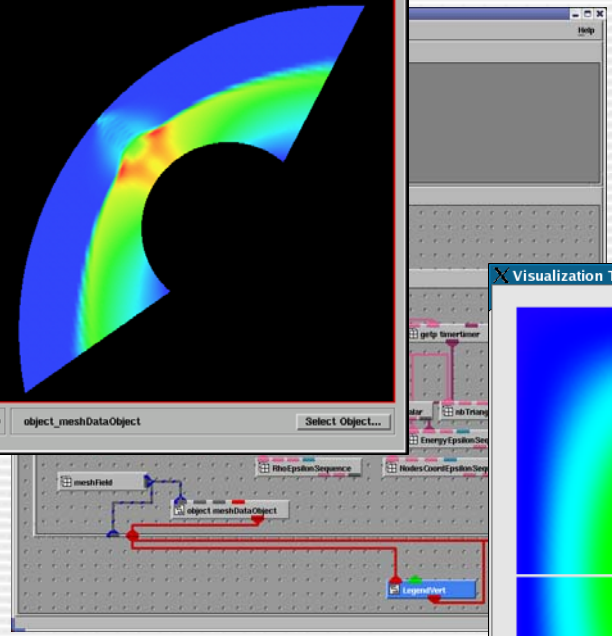
- Distributed and dynamic infrastructure
  - client/server relationship
  - several remote clients can connect and disconnect on-the-fly
  - concurrent client-directed requests
- Instrumentation of legacy simulation
  - annotations of the sources with EPSN C++ API (C/Fortran bindings)
  - multiple instrumentation points
- Communication infrastructure based on CORBA
  - support of hetereogeneous distributed system
  - CORBA server encapsulated in a thread
  - use of CORBA fully hidden to the end-user

Overview of EPSN Architecture

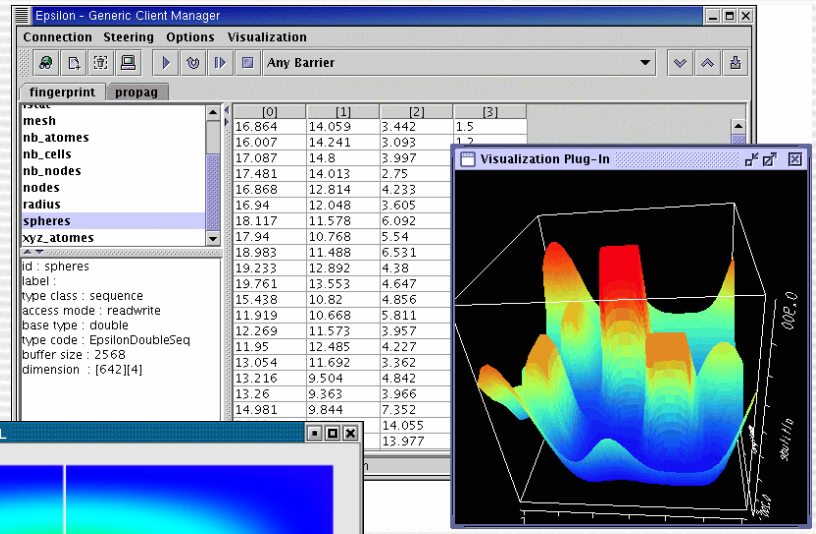


User Interfaces

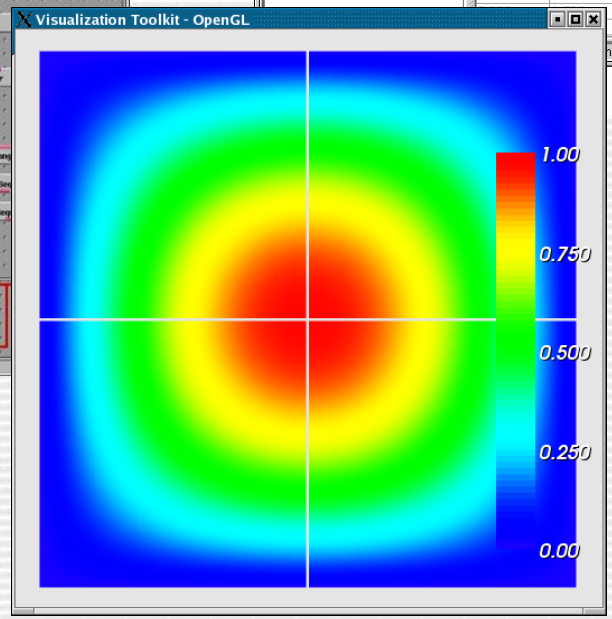




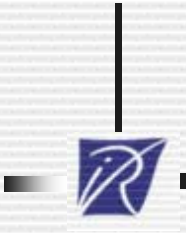
AVS/Express



EPSN Generic Client



Parallel VTK



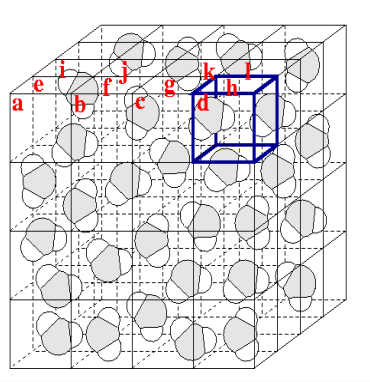
High-Level Model of Steerable Simulations

Introduction

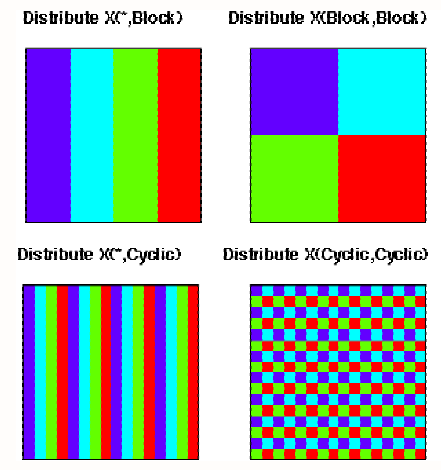
- Representation of the simulation in an abstract model
  - description of the simulation data
  - description of the execution flow as a hierarchy of tasks
- The high level model indicates to the steering system where, when and how it can safely interact with the simulation.
- It also allows us to consider complex simulations (multiple and nested loops).

Data Model

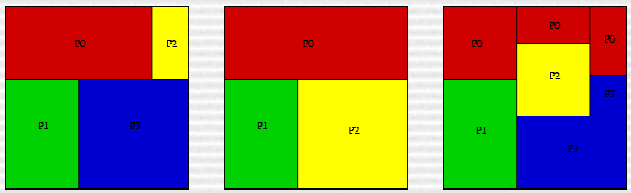
- Complex data objects
  - scalars, fields, particles, unstructured meshes
- Several data series in one object
- Complex storage
  - can match user storage (strides, structures …)
- Block distribution



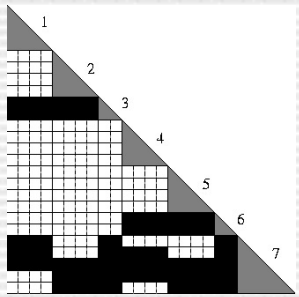
particles in R^3



block-cyclic distribution



rectilinear distribution



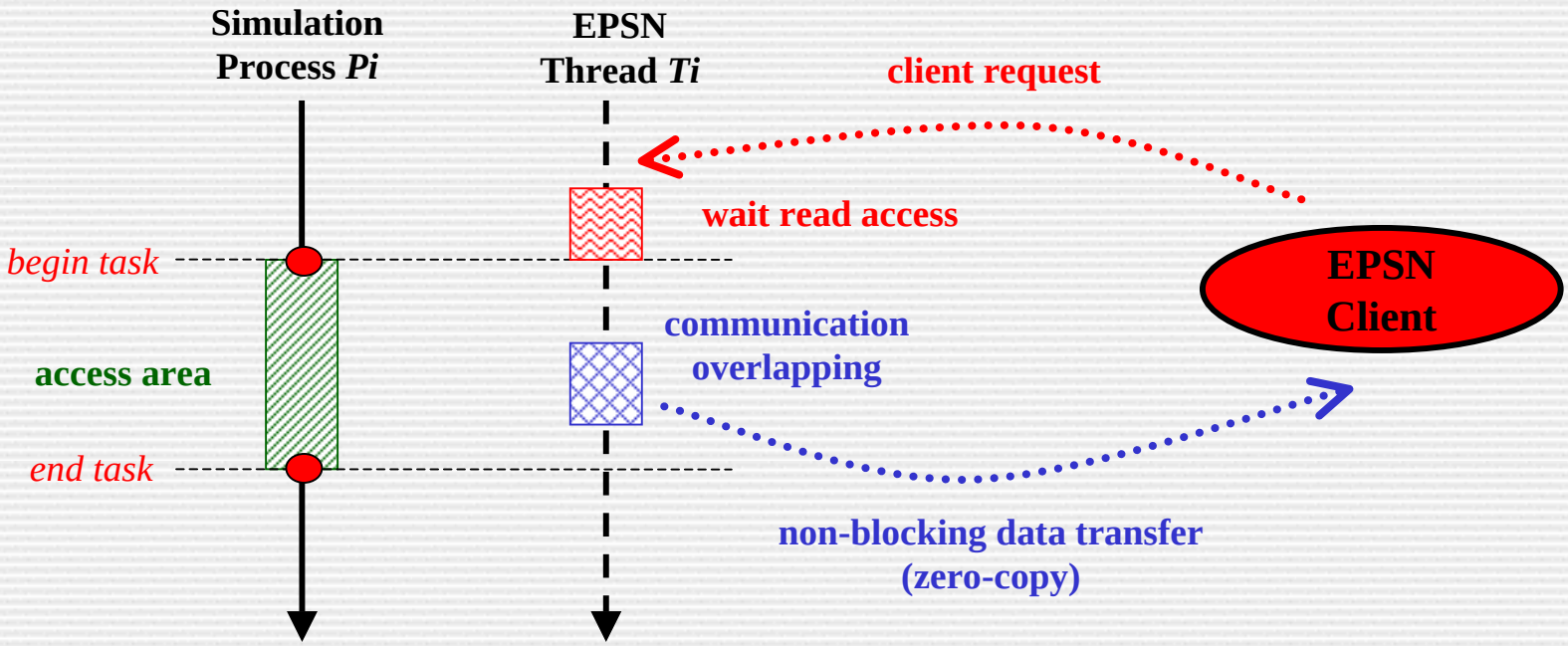
sparse block-matrix

Hierarchical Task Model

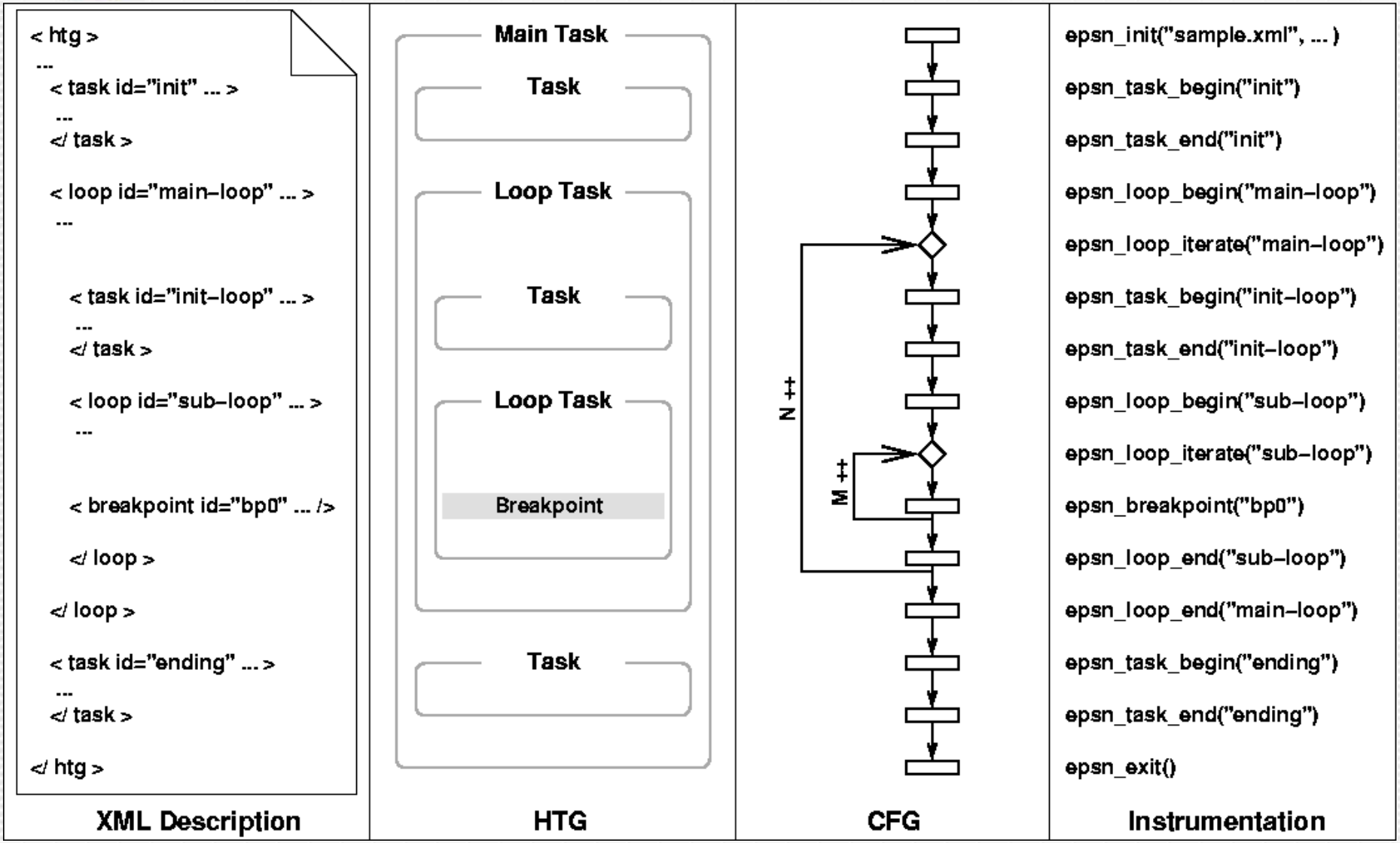
- Based on Hierarchical Task Graph (HTG)
  - Girkar & Polychronopoulos (1994)
  - directed acyclic graph that captures the control flow
- Tasks are hierarchical as they can contain other tasks
- The different hierarchical tasks we consider
  - basic task (block of codes)
  - loop task (while, for)
  - conditionnal task (if, switch)
  - interactive node (breakpoint, action point, writing point)

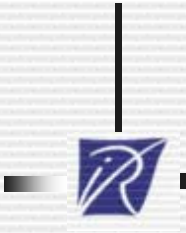
Hierarchical Task Model

- Data access
  - specify if a data is **accessible**, **protected** or **modified** through a task context
  - partially automate the intrumentation process (access area & data release)



Simulation Model & Instrumentation





The Time-Coherence Problem

The Time-Coherence Problem

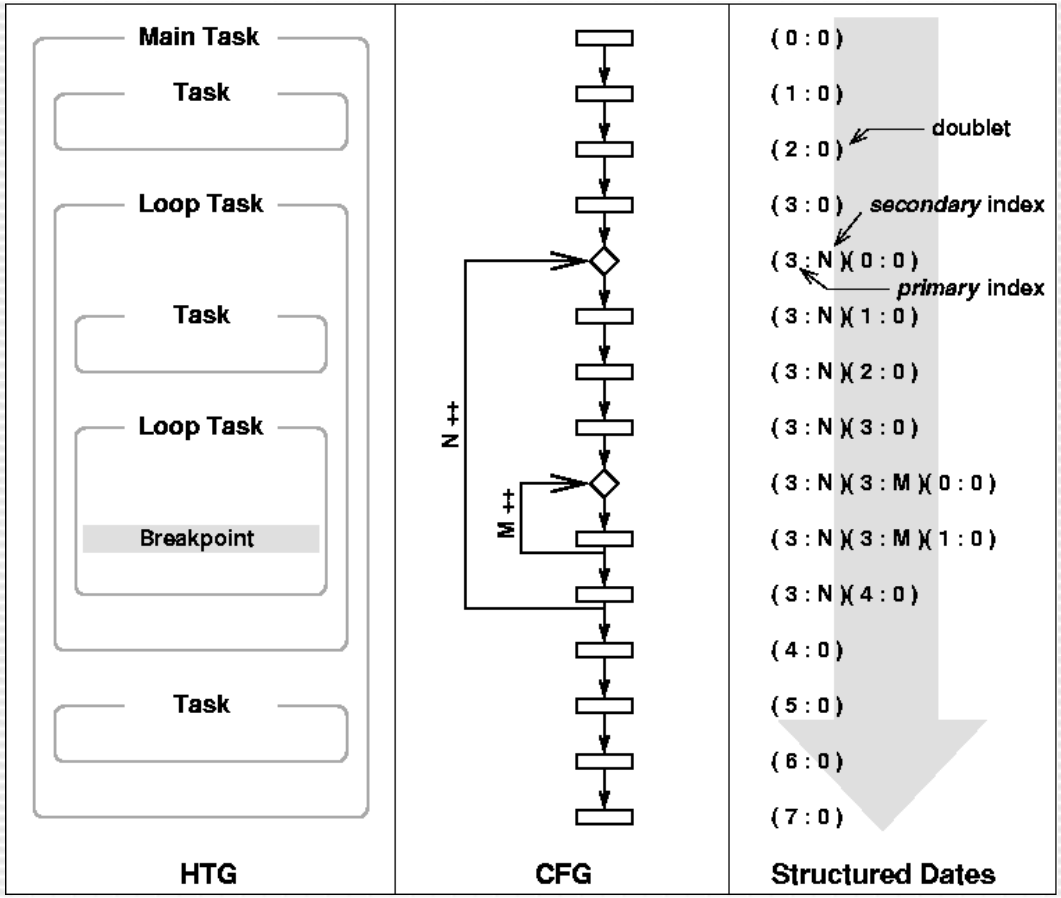
- The on-line visualization and the computational steering of parallel simulations come up against a serious coherence problem.

- Two critical examples
  - distributed data must be accessed carrefuly to ensure they are presented to the visualization system in a meaningful way
  - an action (e.g. checkpoint, modification of a parameter) must occur at the same time for all simulation processes

$\Rightarrow$ How to ensure that the steering treatment will occur at the same time in the simulation?

Structured Dates

We introduce the structured date to precisely follow the time evolution of the simulation in the HTG.

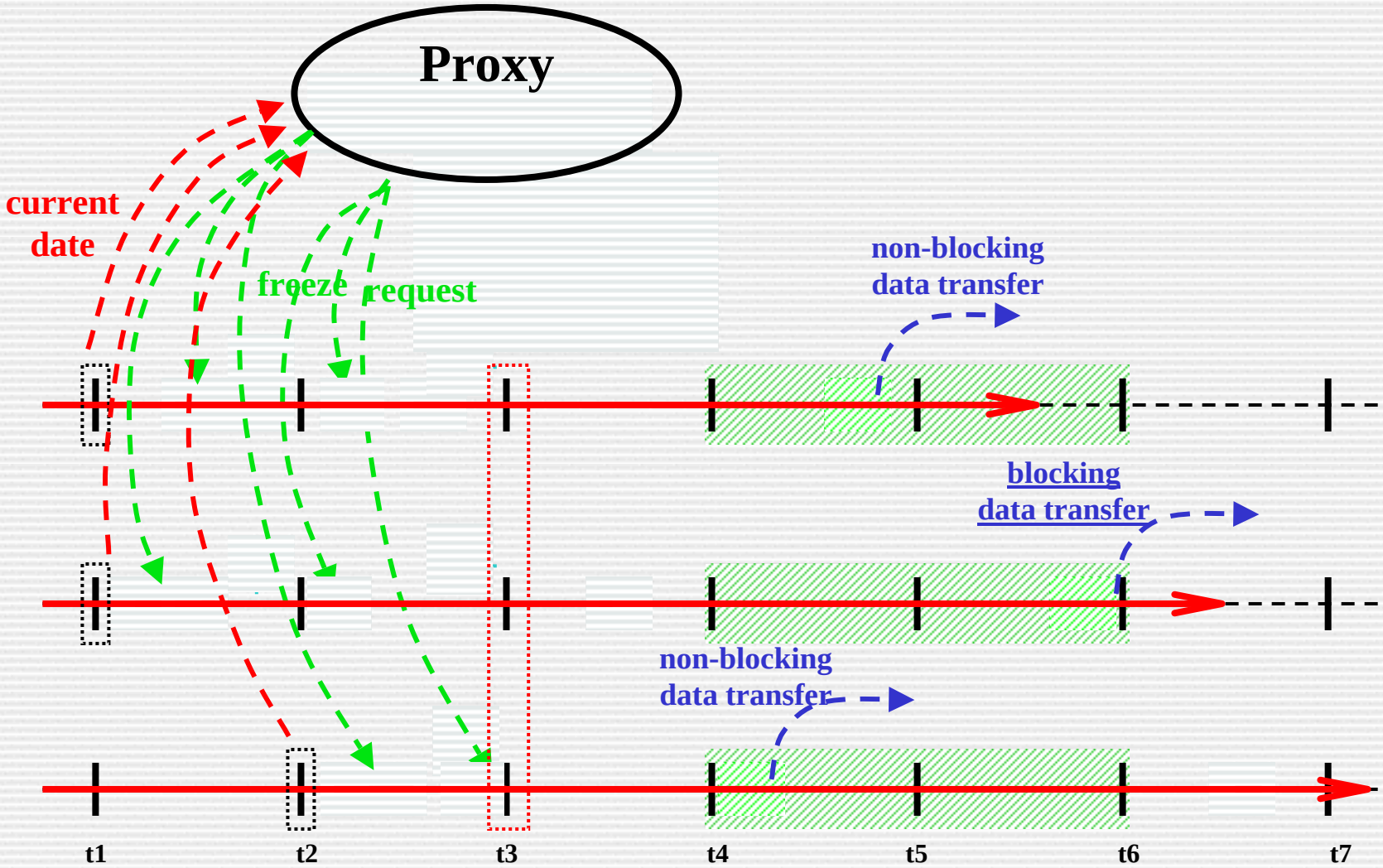


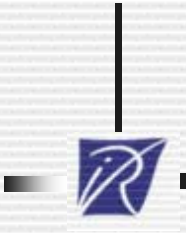
Coherence Strategies

- Post-collection ordering (Magellan/Falcon, …)
  - systematic sending without coherence guarantee (sensor)
  - collection and ordering of data on a central server
  - adapting for monitoring (small data) not for online visualization
- Centralized management (DAQV, …)
  - processes refers to a master process on each instrumentation point
  - lot of communications, strong synchronization
- Scheduled treatment (CUMULVS, EPSN)
  - calculation of a common date for executing the request
  - each process do the treatment independantly when the date is reached
  - no request $\Rightarrow$ no communication, very few synchronization

  $\Rightarrow$ How to calculate the common date?

Scheduling of a Request in Parallel

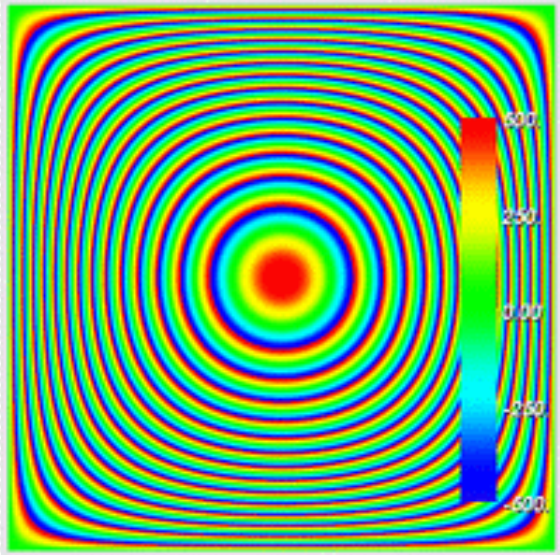




Preliminary Results

2D Heat Equation

- Overview
  - diffusion code (based on Laplace equation)
  - MPI master/slave code (single loop)
  - field 2000×2000 (30.5 MB)
  - 1D block-row distribution
  - small access area
  - data extraction >> computation



Parallel VTK Viewer
1D block-column distribution

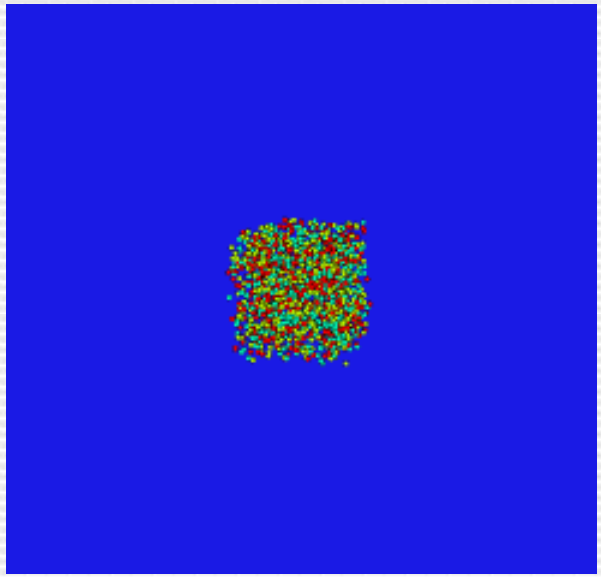
- Preliminary results (in ms/it.)

Simu. × Viewer	***no instr.***	***instr.***	***freq. 25***	***freq. 5***
4 × 1	27,78	27,95	35,49	79,28*
4 × 2	27,78	27,95	32,80	93,40*

*** no overlapping !**

Interacting Particles

- Overview
  - motion of charged particles that interacts electrostatically with each other (Z. Meglicki)
  - MPI code (single loop)
  - 2000 particles (47kB) in strided buffers
  - large access area
  - computation >> data extraction



Parallel VTK Viewer

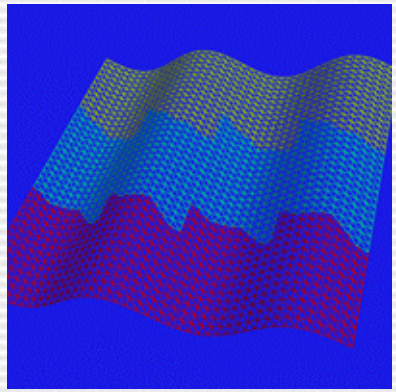
- Preliminary results (in ms/it.)

Simu. × Viewer	***no instr.***	***instr.***	***freq. 1*** ***lock access***	***freq. 1*** ***unlock access***
4 × 1	130,20	130,25	175,93	131,35*
4 × 2	130,20	130,25	163,30	131,85*

*** communication overlapping**

Conclusion & Prospects

- High-level model based on HTG
  - to better grasp the complexity of simulations
  - to closely follow the time evolution (structured date)
- Efficient and time-coherent treatment of parallel request
  - weak synchronization
  - overlapping of data transfers

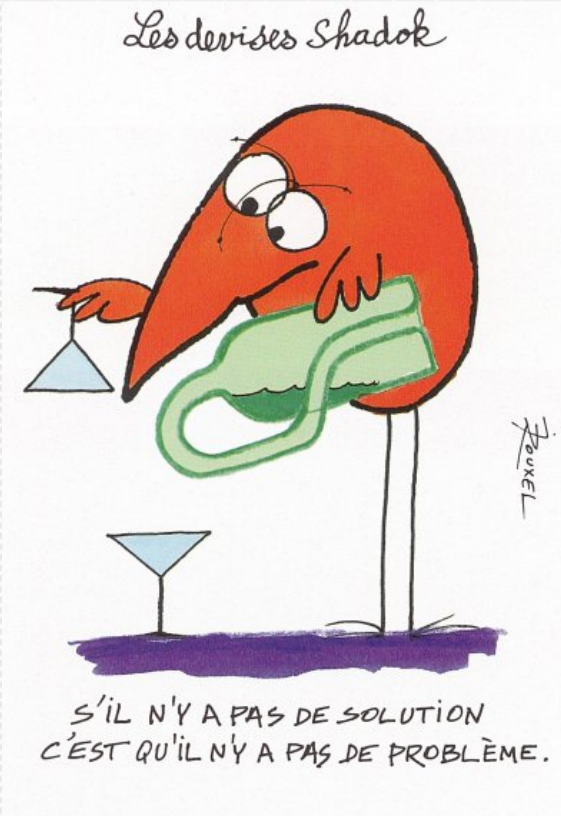


Parallel Mesh Redistribution
our last result

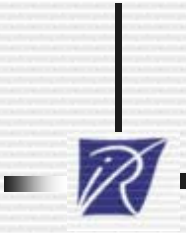
- Prospects
  - extension of our hierarchical model for distributed simulations
  - definition of high-level interaction objects (by direct image manipulation)
  - real-life applications
    - fluid dynamics, dynamic crack propagation, *etc.*

Questions ?

Thank you for your attention.



« If there is no solution, it's because there is no problem. »



Appendix

What is Computational Steering ?

- J. Mulder, J. Wijk, R. Liere – 1999.

« Computational steering can be defined as the **interactive** control over a computational process. »

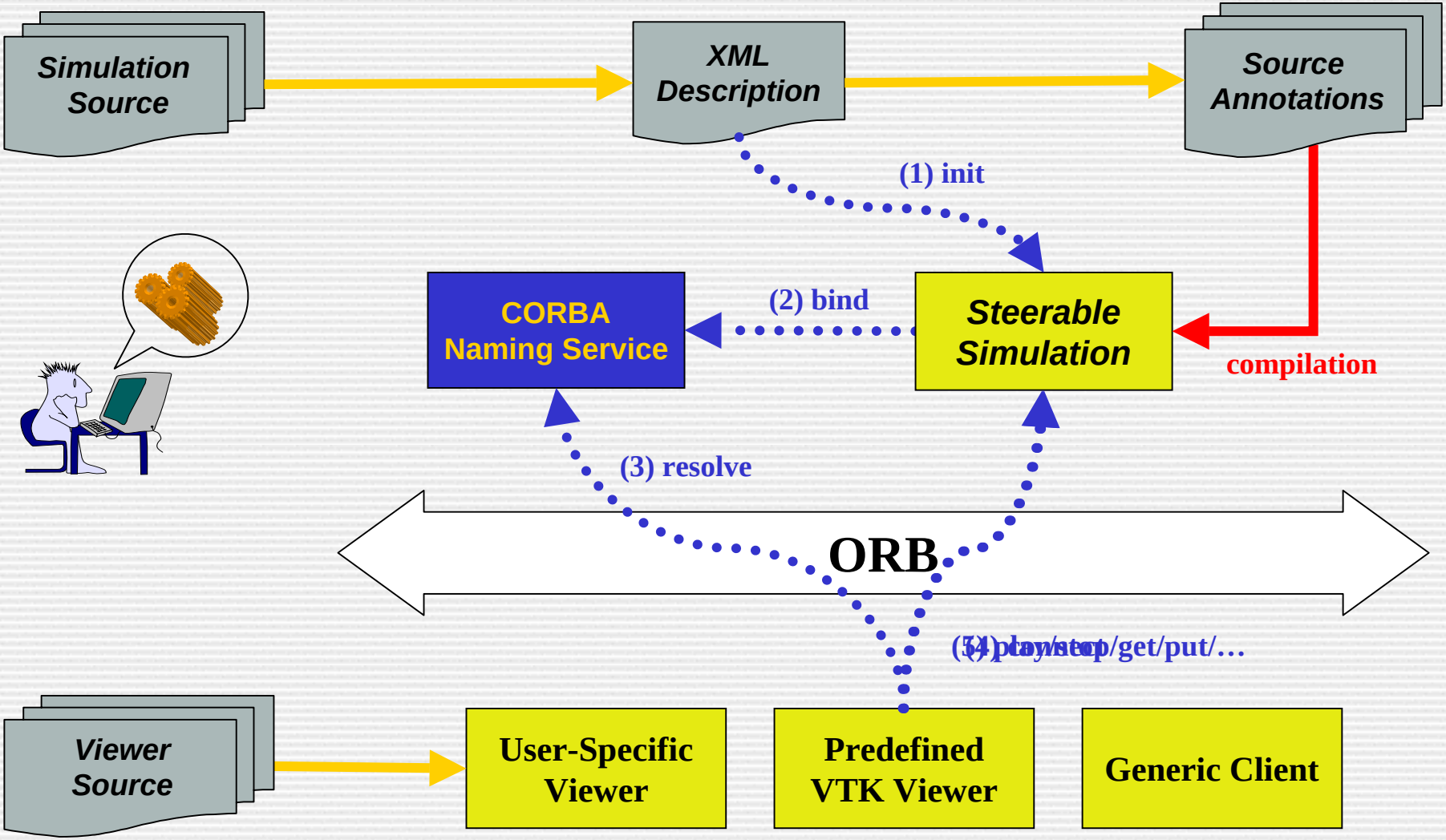
- J. Vetter, K. Schwan – 1996.

« Computational steering is the **run-time** control of an application and of ressources it uses for purposes of experimenting with application parameters or improving application performance. »

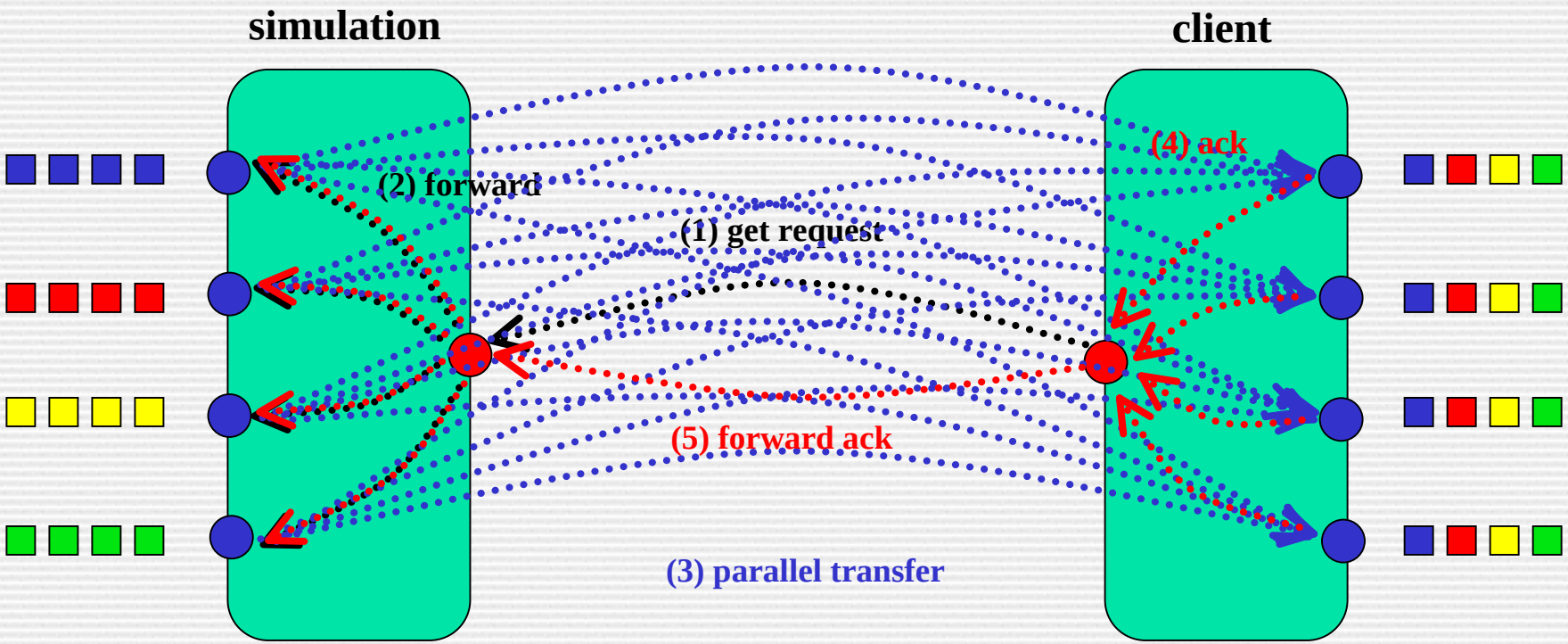
- S. Parker, D. Beazley, C. Johnson – 1996.

« Computational steering (…) allows the efficient extraction of scientific information and permits changes to simulation parameters and data in a **meaningful** way. »

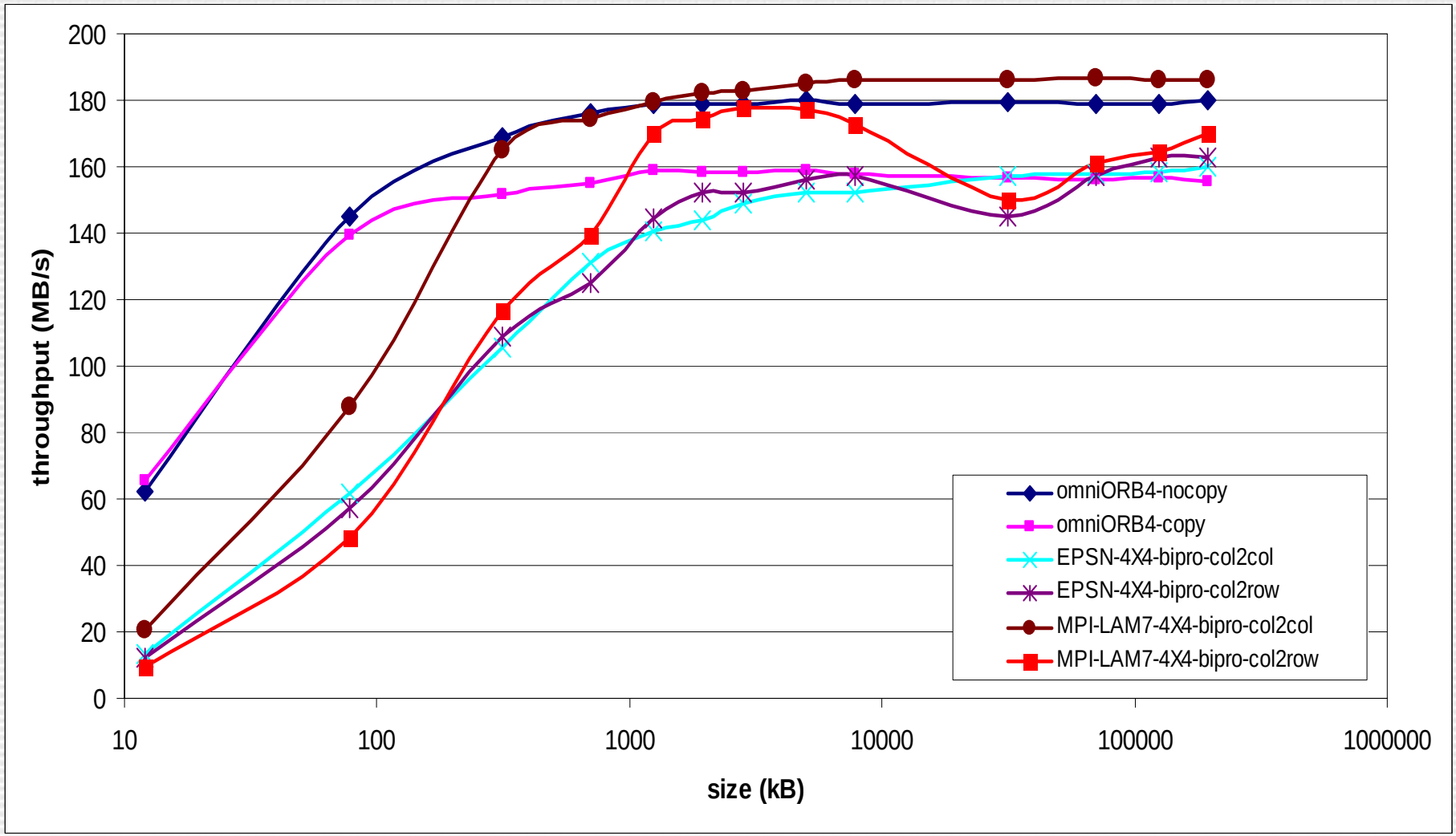
Instrumentation Process + Deployment



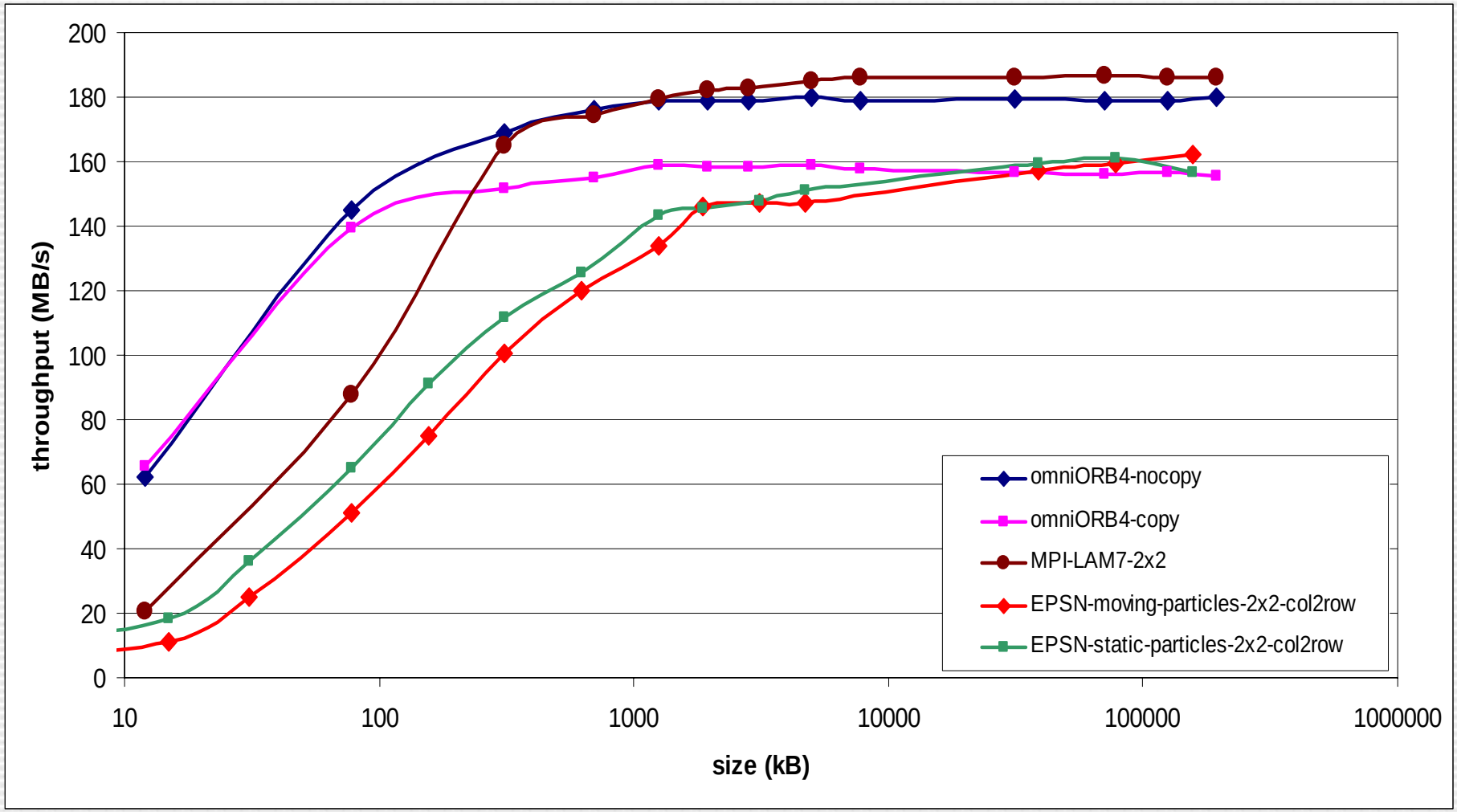
Communication Scheme of the Get Request



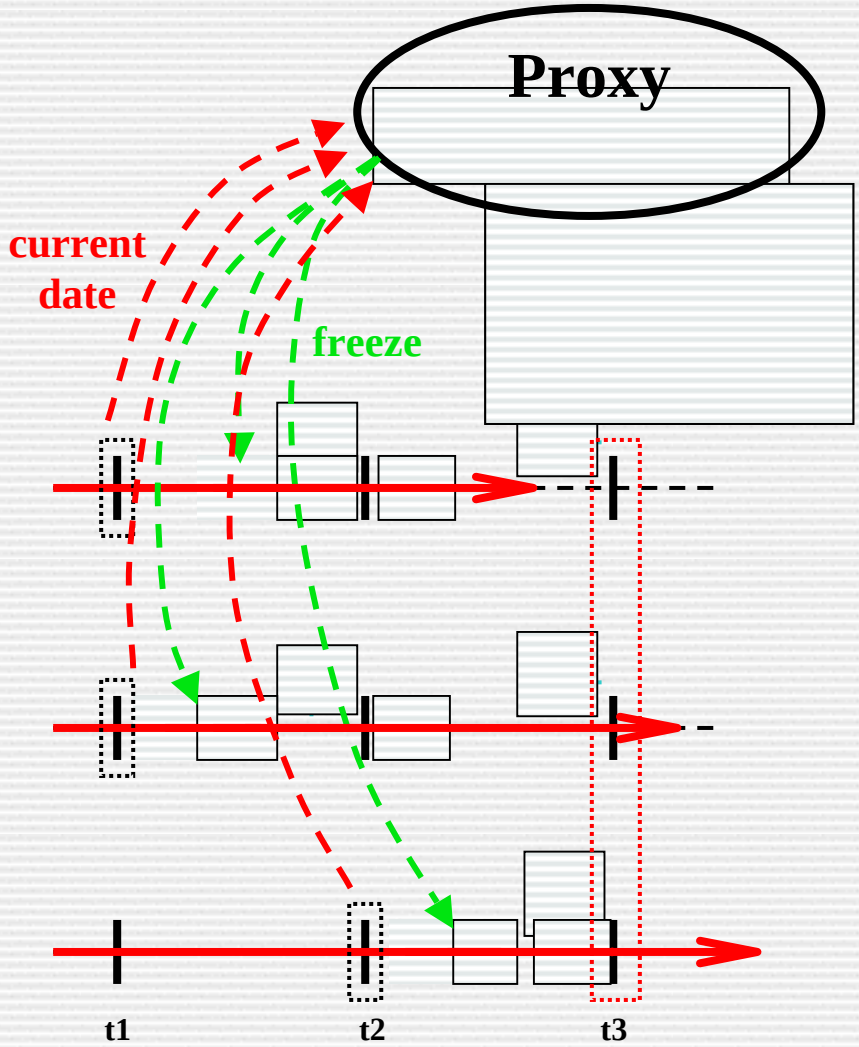
Preliminary Results – Redistribution of Field



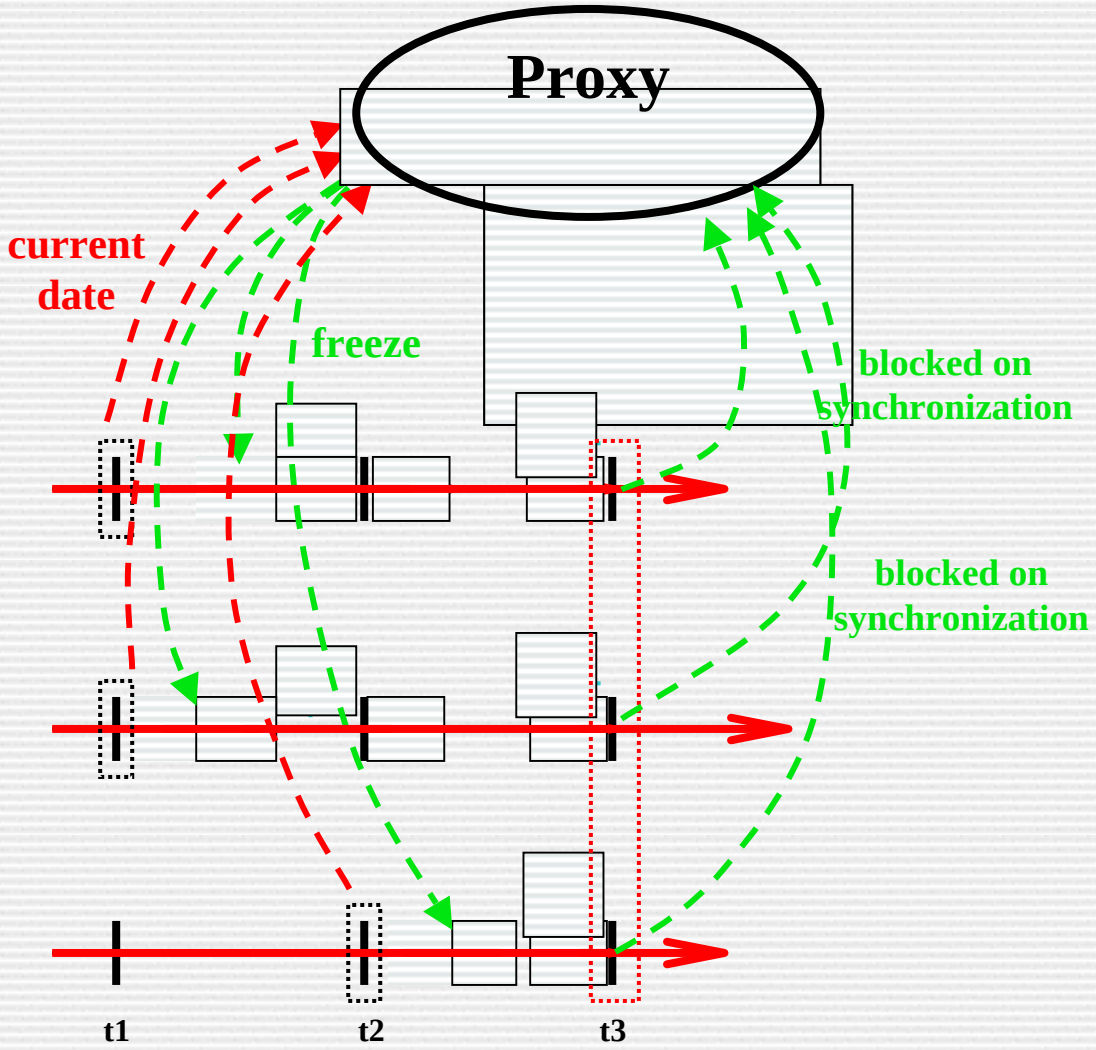
Preliminary Results – Redistribution of Particles



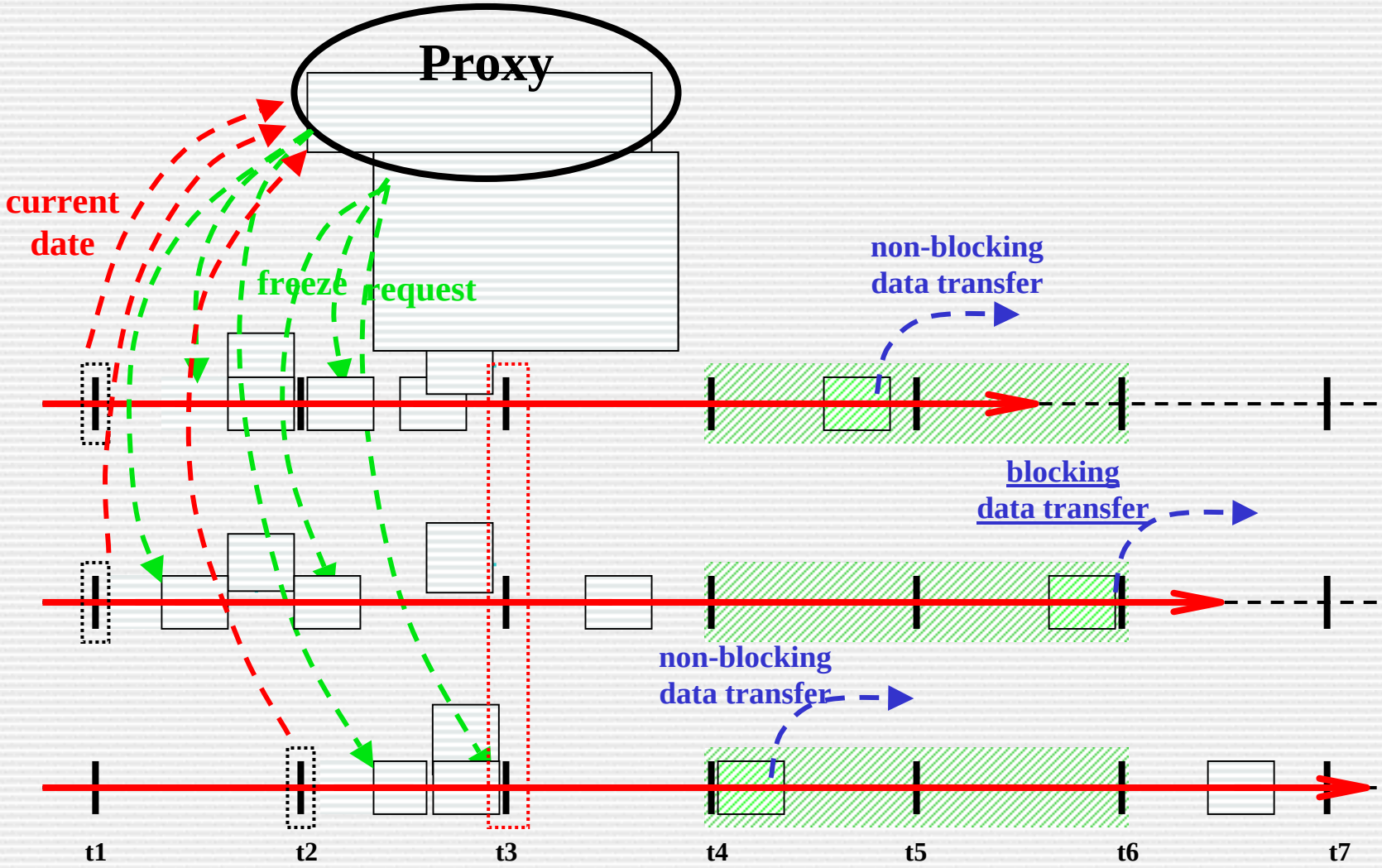
Weak Synchronization



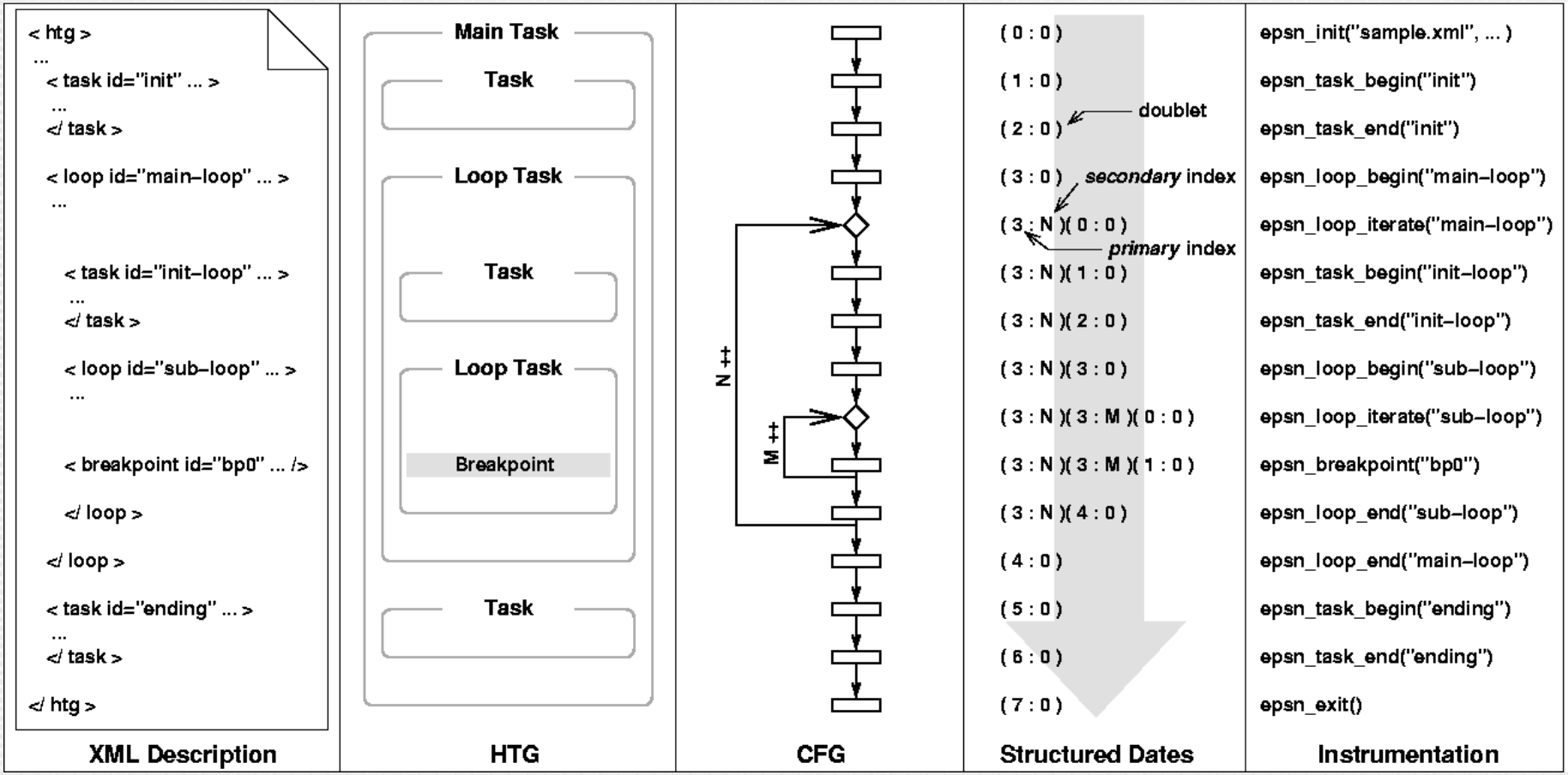
Strong Synchronization



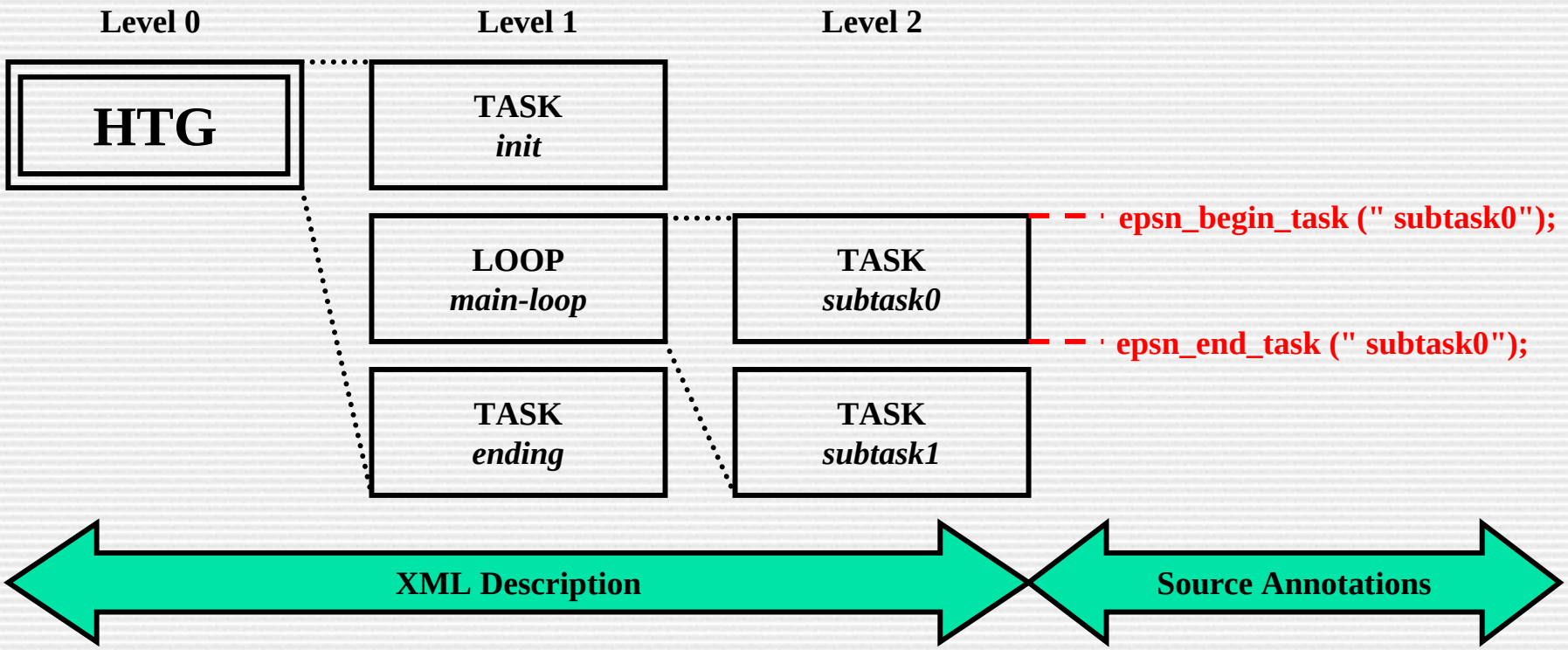
Scheduling of the Get Request



Example


< htg >
...
< task id="init" ... >
...
</ task >
< loop id="main–loop" ... >
...
< task id="init–loop" ... >
...
</ task >
< loop id="sub–loop" ... >
...
< breakpoint id="bp0" ... />
</ loop >
</ loop >
< task id="ending" ... >
...
</ task >
</ htg >
XML Description
Main Task
Task
Loop Task
Task
Loop Task
Breakpoint
Task
HTG
N ‡
M ‡
CFG
(0 : 0)
(1 : 0)
(2 : 0)
doublet
(3 : 0)
secondary index
(3 : N)(0 : 0)
primary index
(3 : N)(1 : 0)
(3 : N)(2 : 0)
(3 : N)(3 : 0)
(3 : N)(3 : M)(0 : 0)
(3 : N)(3 : M)(1 : 0)
(3 : N)(4 : 0)
(4 : 0)
(5 : 0)
(6 : 0)
(7 : 0)
Structured Dates
epsn_init("sample.xml", ...)
epsn_task_begin("init")
epsn_task_end("init")
epsn_loop_begin("main–loop")
epsn_loop_iterate("main–loop")
epsn_task_begin("init–loop")
epsn_task_end("init–loop")
epsn_loop_begin("sub–loop")
epsn_loop_iterate("sub–loop")
epsn_breakpoint("bp0")
epsn_loop_end("sub–loop")
epsn_loop_end("main–loop")
epsn_task_begin("ending")
epsn_task_end("ending")
epsn_exit()
Instrumentation

HTG


Level 0
Level 1
Level 2
HTG
TASK
init
LOOP
main-loop
TASK
subtask0
TASK
ending
TASK
subtask1
epsn_begin_task (" subtask0");
epsn_end_task (" subtask0");
XML Description
Source Annotations

Efficient Data Transfer