AVALON

Algorithms and Software Architectures for Distributed & High Performance Computing Platforms

Christian Perez

2013, April 24

Avalon Members @ February 1st, 2013

Faculty Members (4 INRIA, 1 CNRS, 2 UCBL, 1 ENSL)

- Eddy Caron, MCF ENS Lyon, HDR (80%)
- Frédéric Desprez, DR1 INRIA, HDR (30%)
- Gilles Fedak, CR1 INRIA
- Jean-Patrick Gelas, MCF UCBL
- Olivier Glück, MCF UCBL
- Laurent Lefèvre, CR1 INRIA
- Christian Perez, DR2 INRIA, HDR, Project leader
- Frédéric Suter, CR1 CNRS

PhD students (9)

- Maurice-Djibril Faye, ENSL/U. G. Berger, Sénégal
- Georges Markomanolis, INRIA
- Mehdi Mohamed Diouri, ENSL
- Sylvain Gault, MapReduce, INRIA
- Anthony Simonet, MapReduce, INRIA
- Ghislain Landry Tsafack, Héméra
- Vincent Lanore, ENSL
- Arnaud Lefray, SEED4C/ENSIB
- Daniel Balouek, CIFRE New Generation SR

Engineers (3+7+1)

- Simon Delamare, IR CNRS (40%)
- Jean-Christophe Mignot, IR CNRS (20%)
- Matthieu Imbert, INRIA SED (40%)
- Sylvain Bernard, CloudPower
- Haiwu He, CloudPower
- Julien Carpentier, XLCLOUD
- François Rossigneux, XCLOUD
- Noua Toukourou, ANR COOP
- Guillaume Verger, SEED4C
- Yulin Zhang Huaxi, SEED4C
- *Laurent Pouilloux (AE Héméra)*

Postdoc (3)

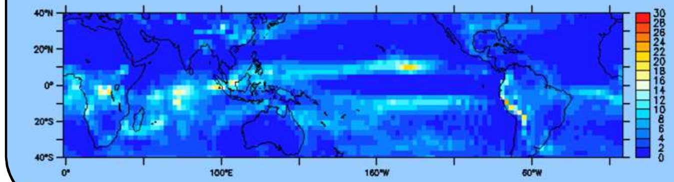
- Zhengxiong Hou, ENS Lyon
- Jonathan Rouzaud-Cornabas, CNRS
- Lamiel Toch, INRIA

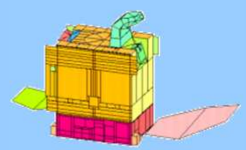
Assistant

- Evelyne Blesle

Avalon: Research Activities

Applications





Programming Abstractions

Algorithmics

Application & Resource Models

Resource Abstractions

?

Elasticity

Energy

Super-computers (Exascale)	Grids (EGI)	Desktop Grids	Clouds (IaaS, PaaS)
Large scale	Heterogeneity	Volatility	On demand

CPU/data-intensive Scientific Applications

- From "simple" to code coupling
  - Structure complexity
  - "New" forms of interactions (MR)

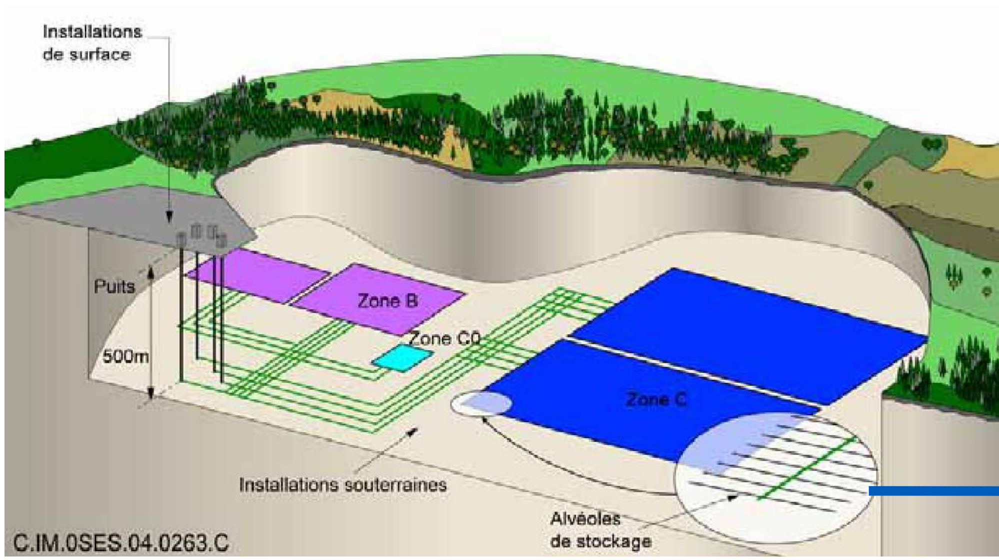
Computing platforms

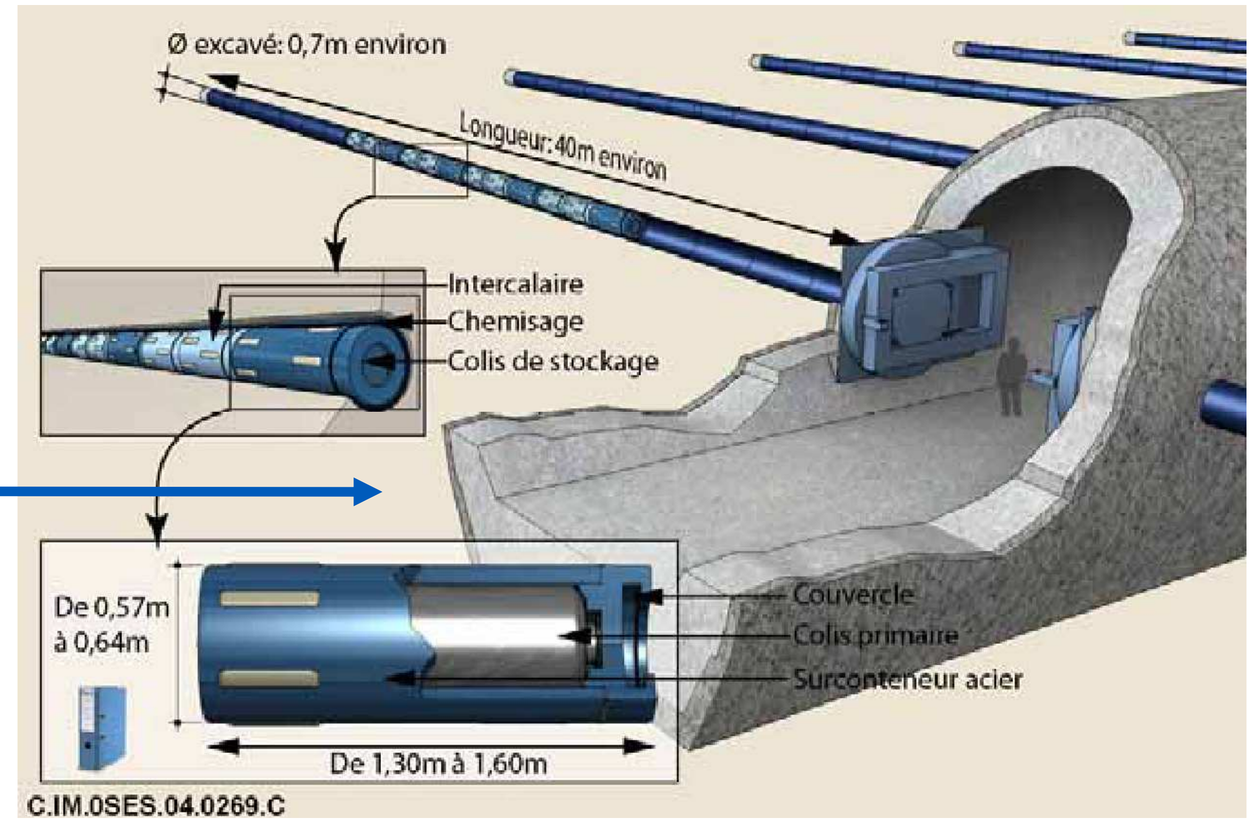
- Different characteristics
  - Performance, energy, size, cost, reliability, QoS, etc.
- Hybridization
  - Sky computing, HPC@Cloud, Exascale, Spot instance

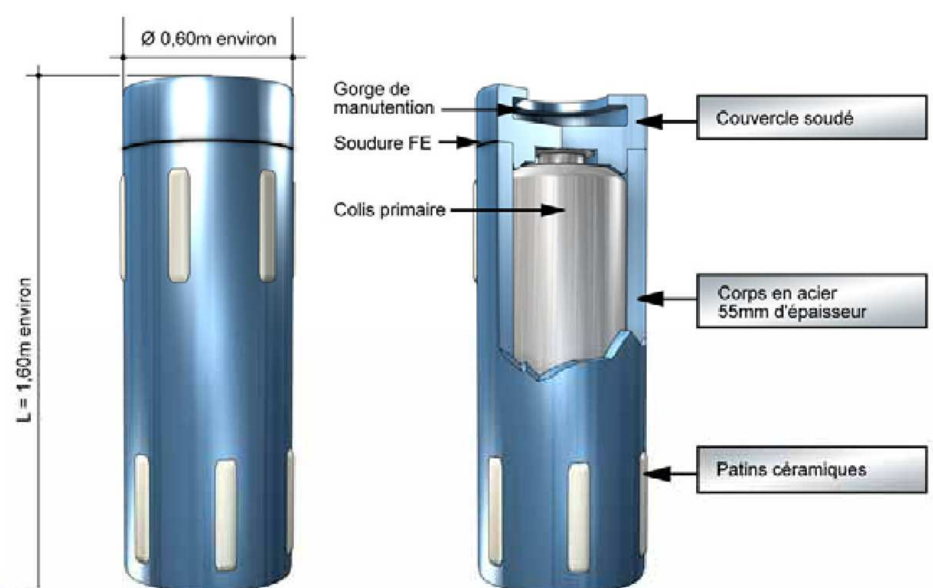
Objectives

- Expressiveness simplicity
- Application portability
- Resource specific optimizations
  - Elastic resource management
  - Energy consumption

Exemple d'ordonnancement: Stockage des CSD-vitrifiés







Traitement de toutes les familles de colis

Familles de colis fictifs :

- Famille Cs^{137}-Sr^{90}, entre 200 et 600 W au moment du stockage
- Famille Cs^{137}-Sr^{90}-Am^{241}, entre 200 et 600 W au moment du stockage
- Famille Cm^{244}, entre 200 et 600 W au moment du stockage
- …

~ 100 colis fictifs différents, plus de 15 000 calculs de thermique

Algorithm 2 Exploration de l'espace des phases pour toutes les familles de colis données

Require: $P_x \in [3.5 : 25]$, $2P_x \in \mathbb{N}$

for famille $\in$ Familles **do**
  for colis $\in$ famille **do**
    for $D_y \in [10, 15, 20, 25, 30]$ m **do**
      for $N_c \in [3 : 18]$ **do**
        Trouve P_x minimal tel que $T_{\mathrm{BG}} < 90^{\circ}C$ et $T_{\mathrm{PdC}} < 100^{\circ}C$
      end for
    end for
  end for
end for

Component Models for HPC

HLCM & L2C

Christian Perez

LIP, ENS Lyon

Programming a Parallel Application

(High level) parallel languages

- HPF, PGAS, …

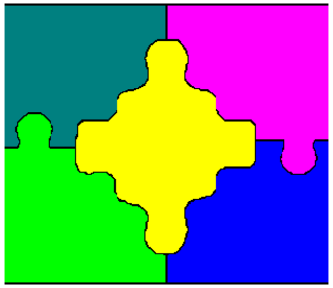
➢ Not yet mature

Platform oriented models

- Multi-core ⇔ Threads, OpenMP
- GPU ⇔ Cuda, OpenCL, OpenAPP
- Multi-node ⇔ MPI

➢ Many versions of the same code
➢ Difficult to maintain all versions synchronized
➢ Difficult to keep specific machine optimizations
➢ Low code reuse

Software Component



Technology that advocates for composition

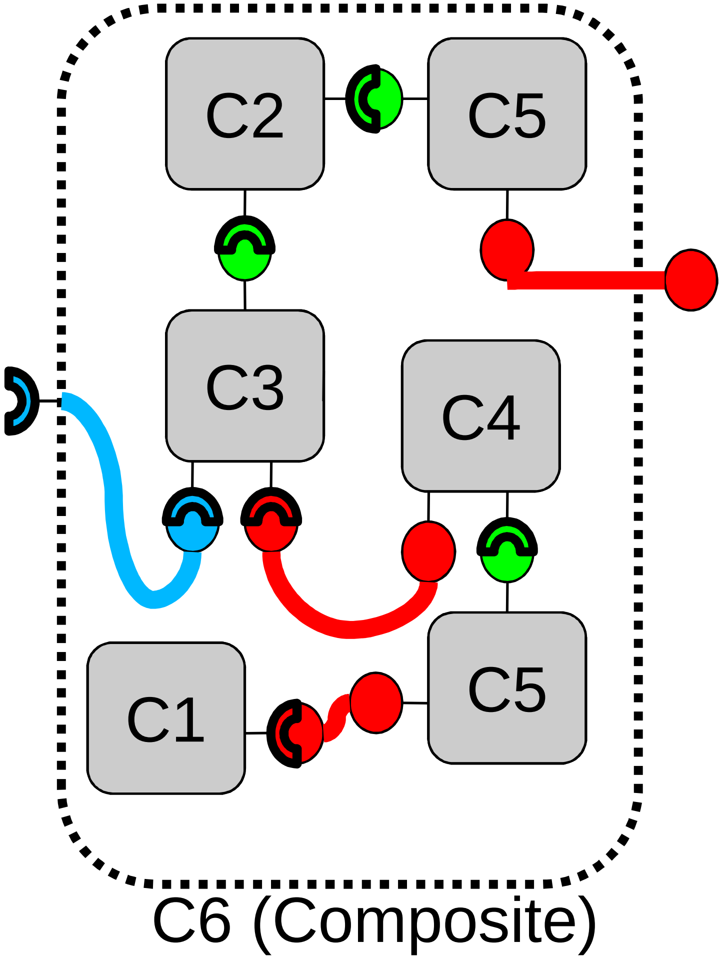
- Old idea (late 60’s)
- *Assembling* rather than *developing*

Many types of composition operator

- Spatial, temporal, …. composition

Assembly of component

- Primitive & composite components



Many models

- Salome, CCA, CCM, Fractal, OGSi, SCA, …

Application in Hydrogeology: Saltwater Intrusion

Coupled physical models

One model = one software

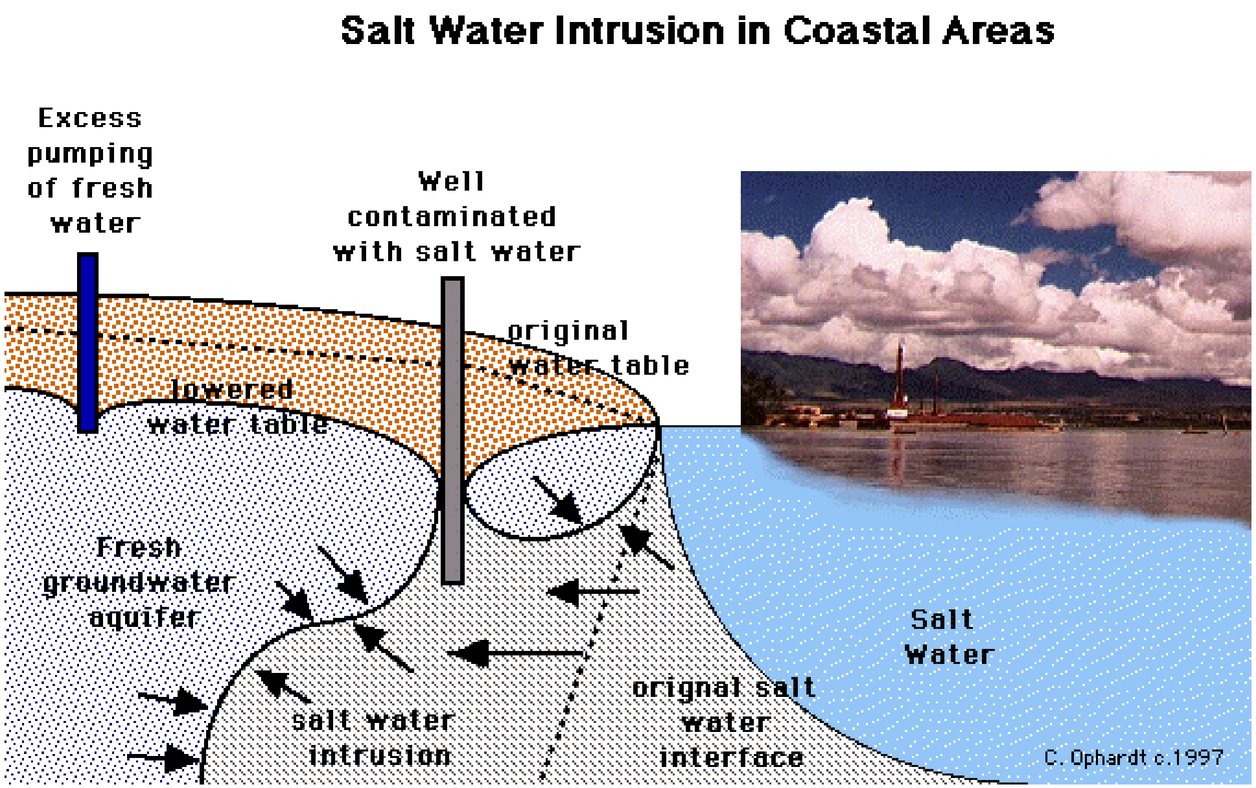
Saltwater intrusion

- Flow / transport

Reactive transport

- Transport / chemistry

Hydrogrid project, supported by the French ACI-GRID

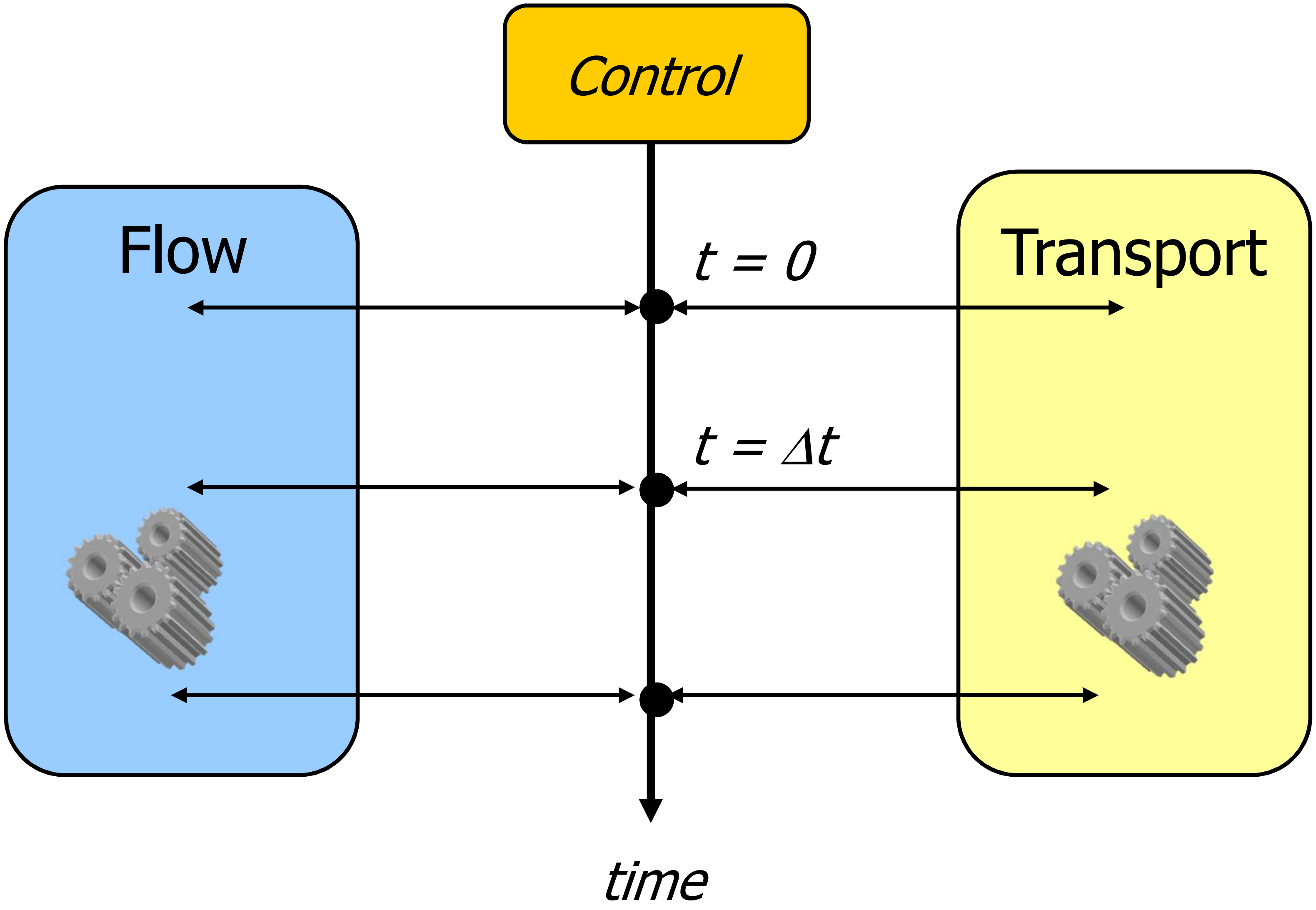


Flow: velocity and pressure function of the density

Density function of salt concentration

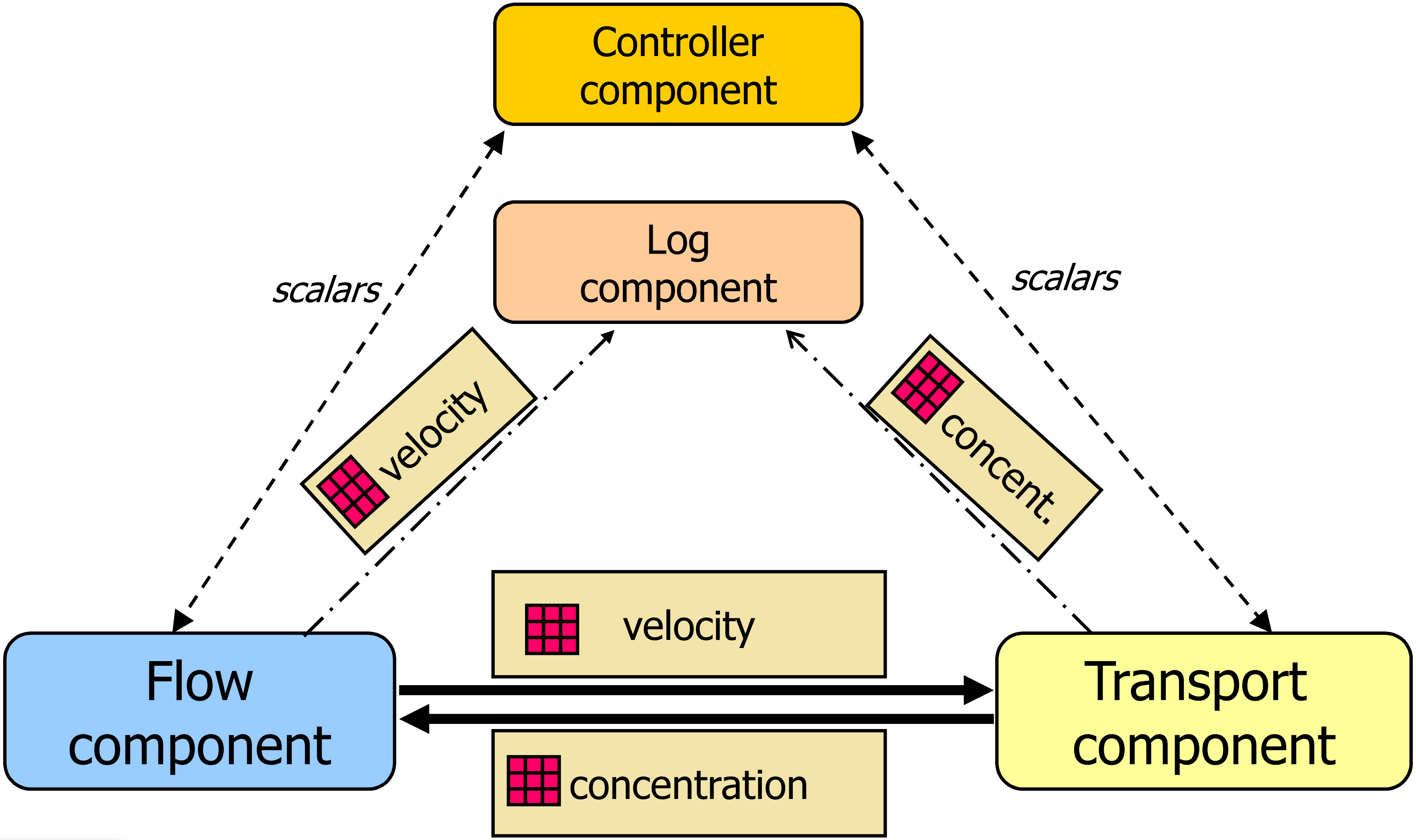
Salt transport: by convection (velocity) and diffusion

Numerical Coupling in Saltwater Intrusion

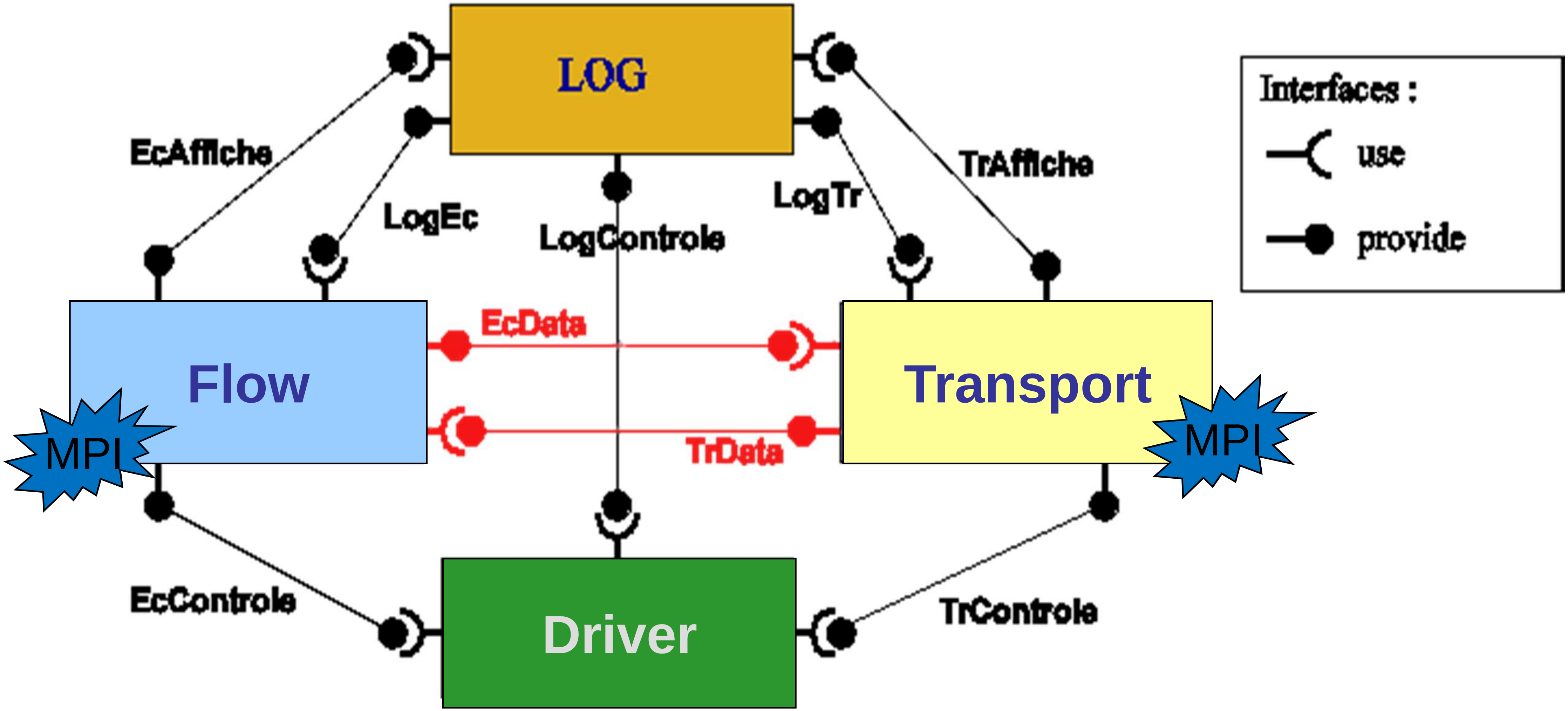


iterative scheme at each time step

Components and communications of PCSI



Components and interfaces of PCSI



Application in Hydrogeology: Saltwater Intrusion

Coupled physical models

One model = one software

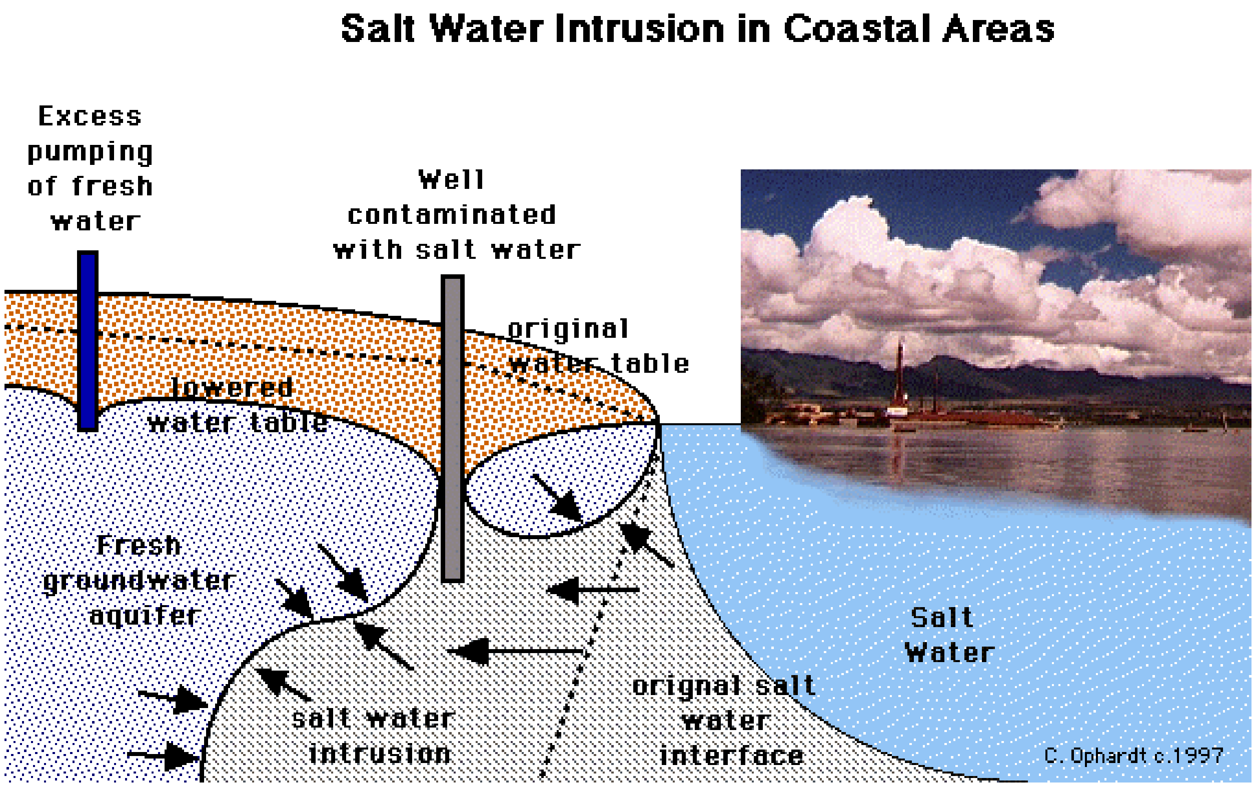
Saltwater intrusion

- Flow / transport

Reactive transport

- Transport / chemistry

Hydrogrid project, supported by the French ACI-GRID

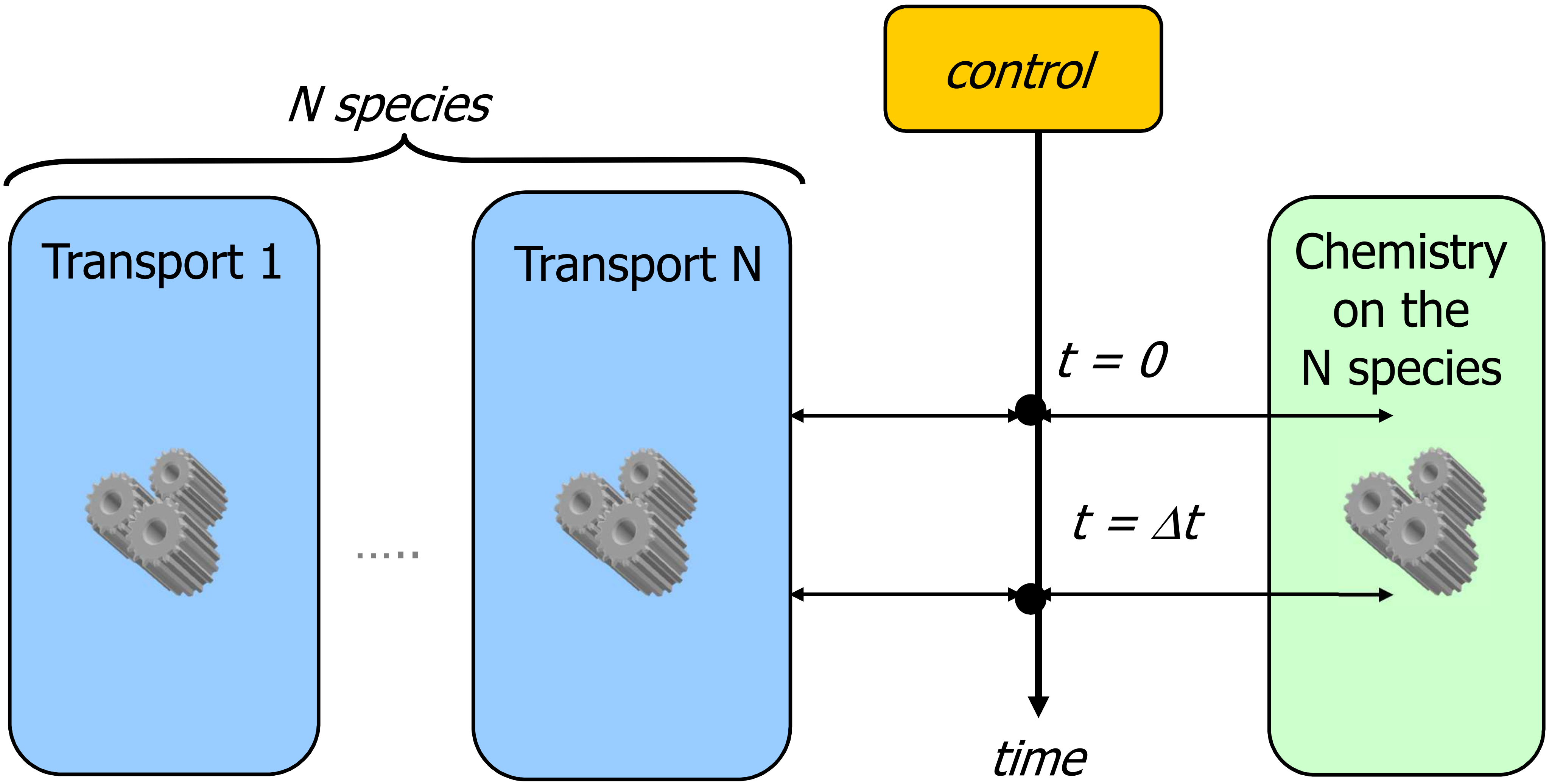


Flow: velocity and pressure function of the density

Density function of salt concentration

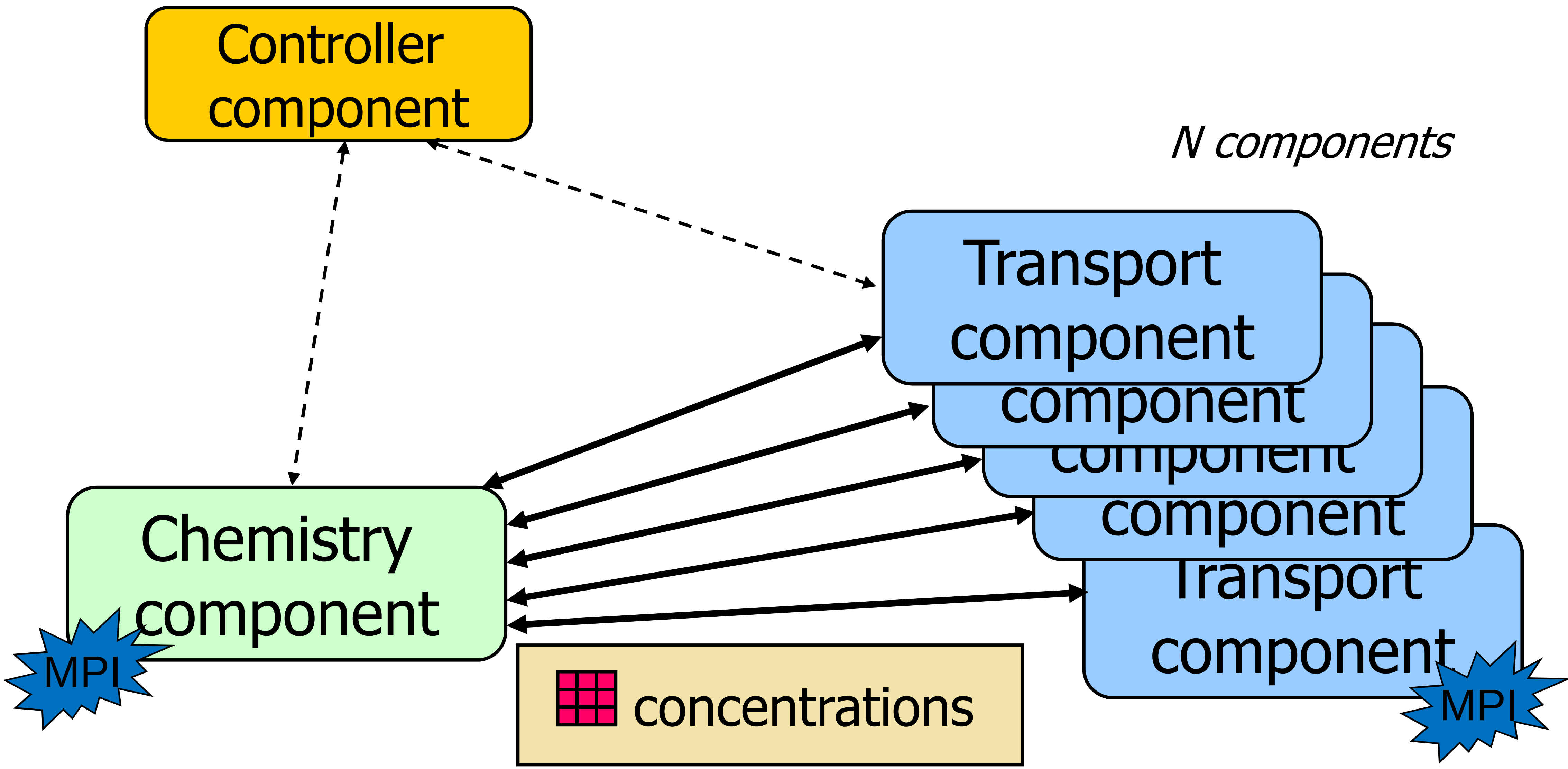
Salt transport: by convection (velocity) and diffusion

Numerical coupling in reactive transport



Iterative scheme at each time step

Component Model for Reactive Transport



Components for Parallel Computing

Memory sharing between components

- CCA & CCM Extensions

Parallel components

- CCA, SCIRun2, GridCCM, Salome

Collective communications

- CCM Extension

Parallel method calls

- SCIRun2, GridCCM, Salome

Master / worker support

- CCA & CCM Extensions

Some algorithmic skeletons in assemblies

- STKM, Salome

Two types of features

- Component **implementations**
  - ≈ skeletons
- Component **interactions**

Limitations of Existing HPC Component Model

Pre-defined set of interactions

- Usually function/method invocation oriented
- ➢ How to incorporate other interactions, e.g. MPI?

Provide communication abstraction

- Language interoperability (~IDL)
- Network transparency
- ➢ Potential overhead when not needed
- ➢ Limited data types systems
  - Babel SIDL, OMG IDL, …

➢ ***Programming model vs execution model***

Objectives

Enable code-reuse

Let expert develop a piece of code

- Software Component
  - Primitive component for re-using implementation code
  - Composite component for re-using assemblies of components

Enable *adaptation* when re-using code

Let re-use code with parameterization options

- Genericity

Enable any kind of composition operators

Do not impose any communication models

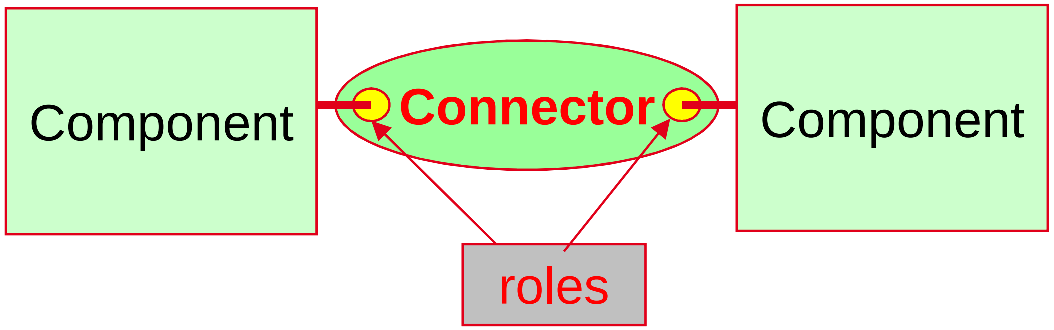
- Connectors

Enable efficient implementation of composition operators

Let have (resource) specific implementations

- Open connection

HLCM: High Level Component Model



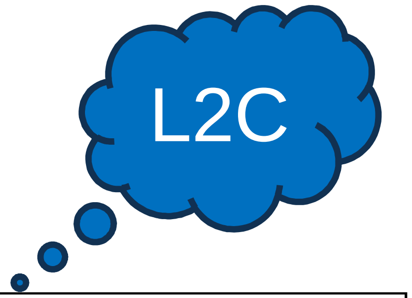
Major concepts

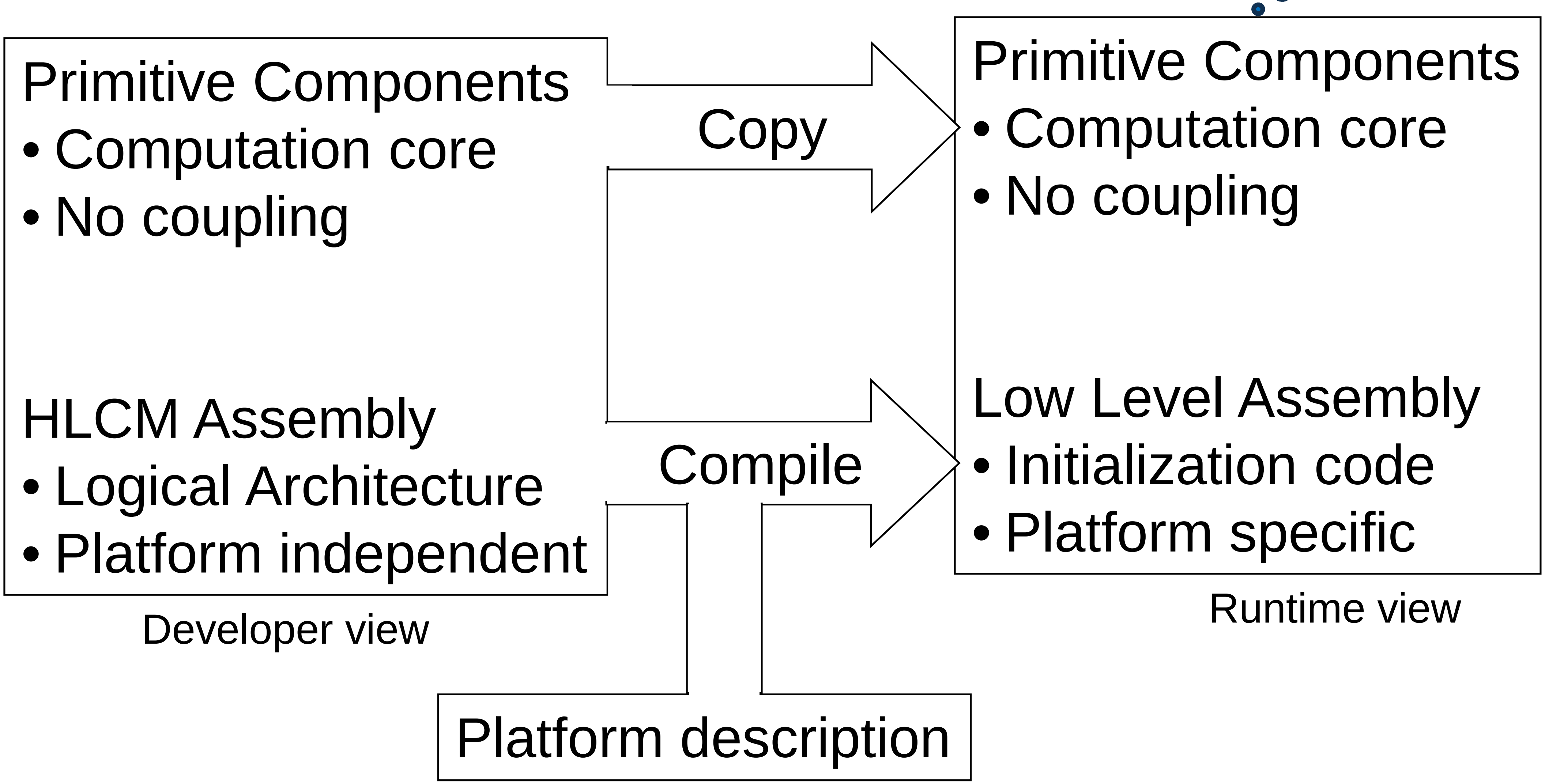
- Component model (hierarchical)
  - Primitive and composite
- Connector based
  - Primitive and composite
- Generic model
  - Support meta-programming (template *à la* C++)
- Currently static

HLCMi: an implementation of HLCM

- Model-transformation based (EMF)
- Connectors
  - Use/Provide
  - Shared Data
  - Collective Communications
  - MxN
  - Some skeletons
    - Replication, Simple Domain Decomposition, MapReduce

HLCM: Overview





L2C: Low Level Component Model

A minimalist component model for HPC

- Component creation/deletion & connection
- An (optional) launcher

No L2C code between components @ runtime

Support native interactions

- FORTRAN procedure, C++ interface, MPI, CORBA

Extensible

LGPL, available at *hlcm.gforge.inria.fr*

Conclusion

Software component is a promising technology for handling code & resource complexity

Component model as a *programming* model

- Many composition operators has been already defined & prototyped
- Re-use existing specialized middleware
- Good feedback from users
- HLCM as a general purpose component model
- L2C as primitive component model

Future work

- Other operators? Finer grain?
  - Domain decomposition, AMR, MapReduce, …
- HLCM expressiveness
  - GPU? PGAS?
  - Temporal composition seems possible
- Automatic component/connector selection & configuration
  - Need to interact with resources