# Overview of Component Model related Activities

## C2S@Exa IPL
## Programming Model Pole

Christian Perez

2015, January 13

# Programming a Parallel Application

## (High level) Parallel Languages

- HPF, PGAS, …

➢ Not yet mature

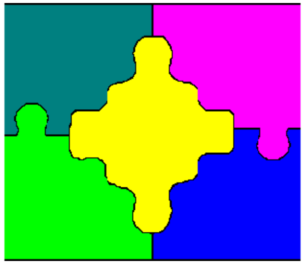
## Platform Oriented Models

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

➢ Many versions of the same code

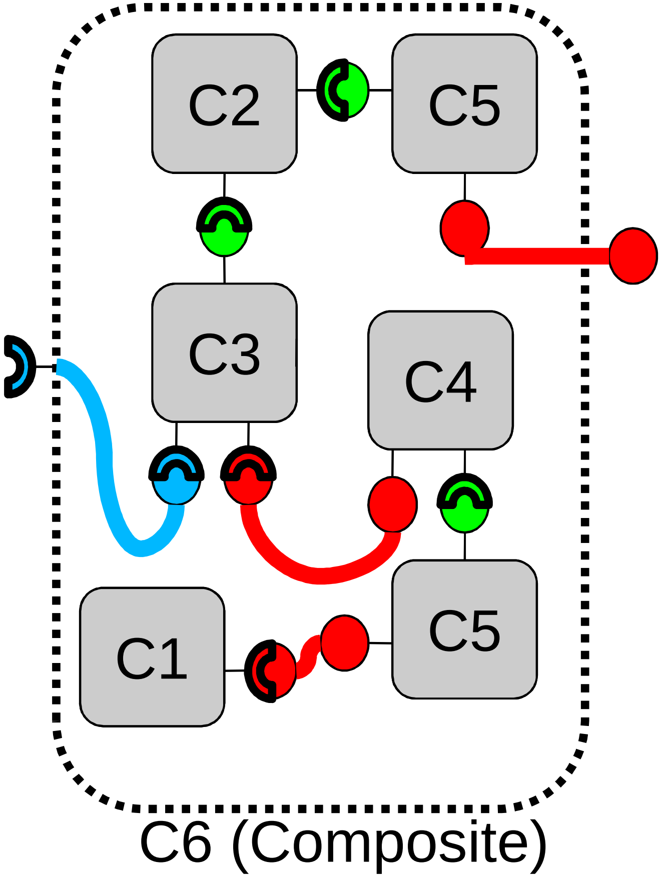
➢ Difficult to maintain all versions synchronized
➢ Difficult to keep specific machine optimizations
➢ Low code reuse

# 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, …

# Motivation for High Level Component Model

**Enable code-reuse**

Let expert develop a piece of code

**Enable *adaptation* when re-using code**

Let re-use code with parameterization options

**Enable any kind of composition operators**

Do not impose any communication models

**Enable efficient implementation of composition operators**

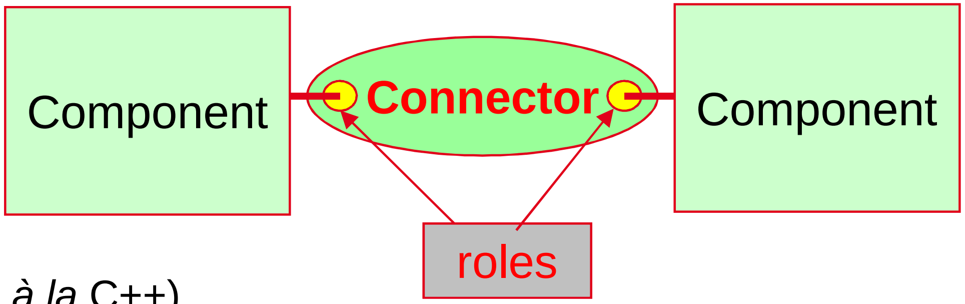
Let have (resource) specific implementations

**PhD of J. Bigot (INSA Rennes, 2010)**

# High Level Component Model (HLCM) Overview

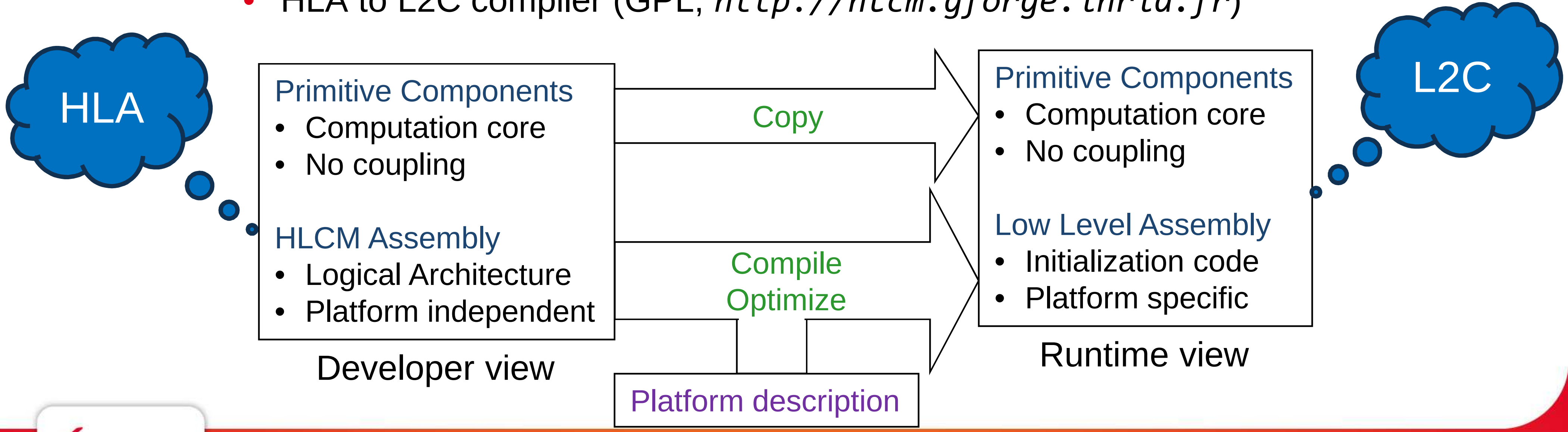
## Major concepts

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



## Implementation

- HLA to L2C compiler (GPL, *http://hlcm.gforge.inria.fr*)



HLA

Primitive Components
- Computation core
- No coupling

HLCM Assembly
- Logical Architecture
- Platform independent

Developer view

Copy

Compile
Optimize

Platform description

Primitive Components
- Computation core
- No coupling

Low Level Assembly
- Initialization code
- Platform specific

Runtime view

L2C

# Low Level Component Model (L2C)

## Developed by J. Bigot & C. Perez

## A minimalist component model for HPC

- Primitive component creation/deletion & connection
- An (optional) launcher

## Features

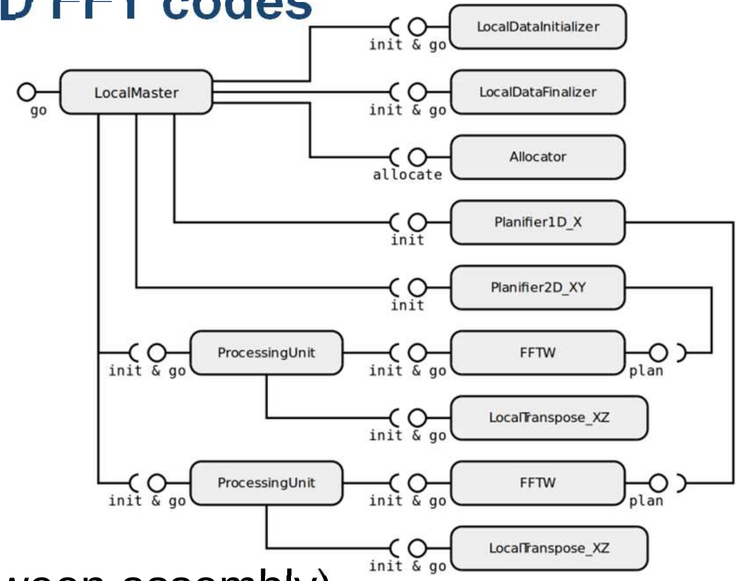
- No L2C code between components @ runtime
- Support native interactions
  - C++, (FORTRAN in progress), MPI, CORBA
- Extensible
- Experiments up to 8,192 MPI processes

**LGPL, *http://hlcm.gforge.inria.fr***

# Designing 3D FFT Assemblies with L2C

## Study how L2C can handle variability of 3D FFT codes

- Assembly adaptation
- Component replacement

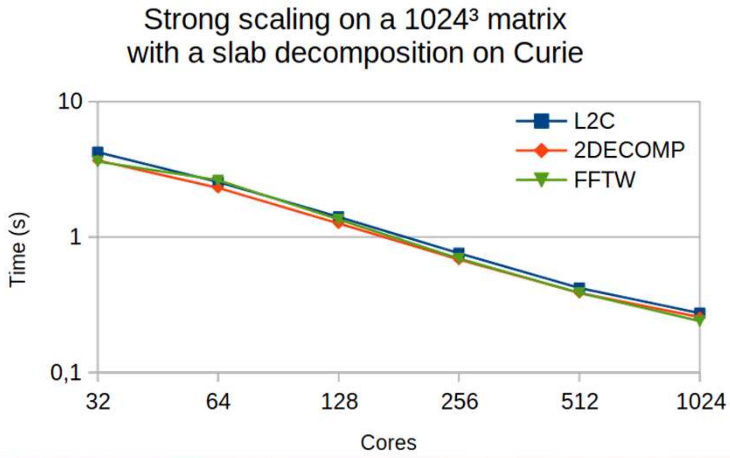


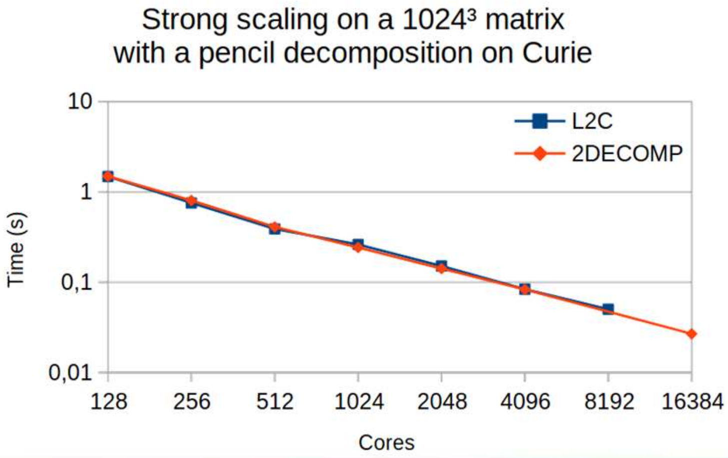
## 3D FFT L2C Implementation

- 1D / 2D domain decomposition
- Load balancing
- Basic overlapping

## Experimental results

- Easy to handle variants (code highly reused between assembly)
- High performance (competitive with reference implementations)


Strong scaling on a 1024³ matrix with a slab decomposition on Curie


Strong scaling on a 1024³ matrix with a pencil decomposition on Curie

# Reconfigurable HPC Component Model: directMod

## Goals

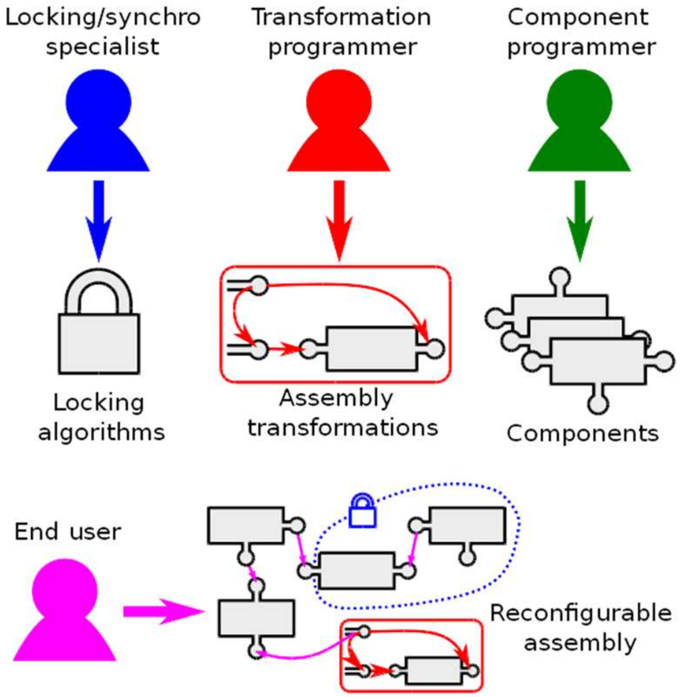
- Scalability
- Separation of concerns
- Reuse

## Principles

- Flat
- Arbitrary locking units (*domains*)
- Standalone transformations
- *Transformation adapters*

## Model

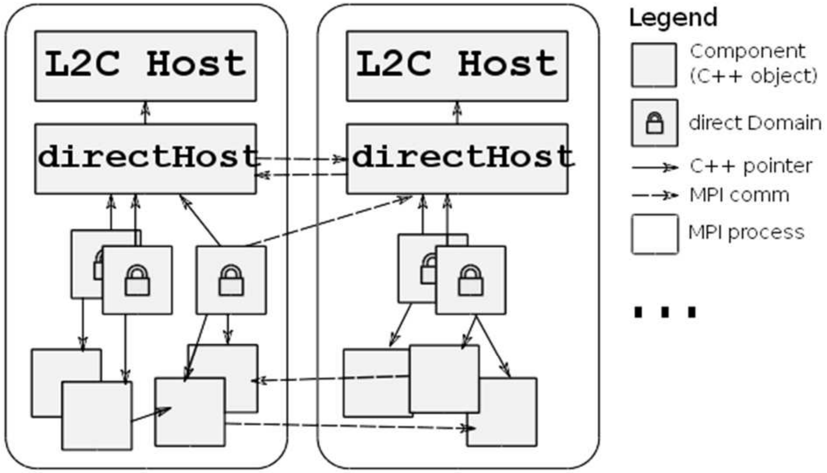
- Full syntax
- Operational semantics
  - Multithread call stacks
- Transformation semantics



## Evaluation at model level

- Self-refining ring (~1D AMR)
  - Easy reuse across variants w/ and w/out work stealing
- 2D AMR
  - ~ 30 different transformations
  - fine control of transformations

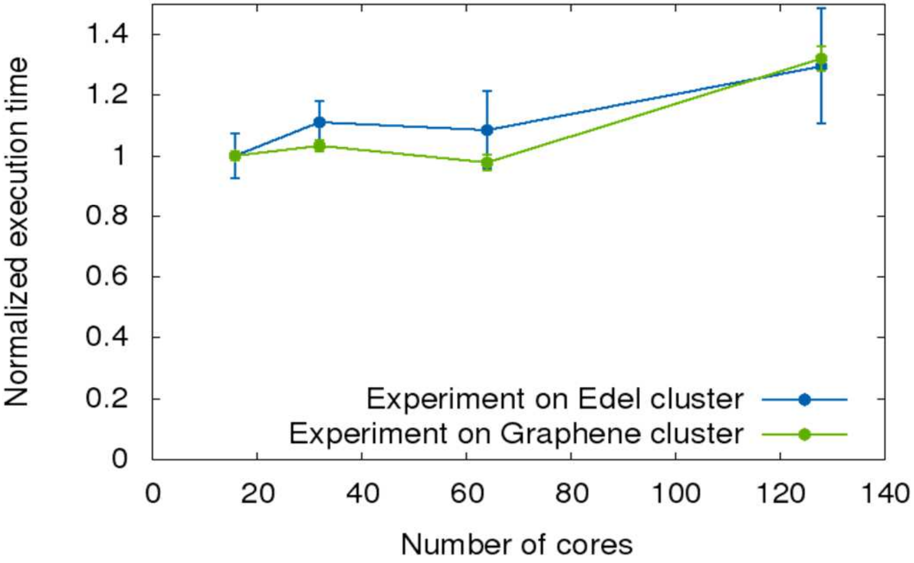
# A directMOD implementation on top of L2C: directL2C



## Preliminary results

- Ring example
  - Simple transformation code (8 lines of C++)
  - Scalability on Grid'5000 up to 128 cores
  - ~5ns static overhead (mutex)

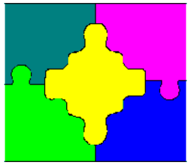
directL2C ring assembly execution time on Grid'5000



## Limitations

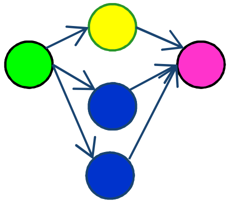
- Multithreaded MPI performance
  - MadMPI
- Locking code error prone

# Component Models and Data Flows



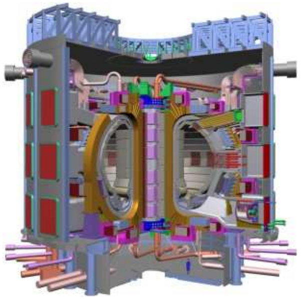
## Two orthogonal concerns

- Structure description: component model (eg, L2C)
- Task dependencies: data flow model (eg, StarPU, XKaapi)



## Goal

- Merge both concerns into a common HPC model
  - Define a L2C+StarPU related model
  - Apply it to Gysela5D / retrieve problematic patterns from Gysela5D
- PhD organization
  - Jérôme Richard’s C2S@Exa PhD  (11/2014-10/2017)
  - Advisor: C. Perez (Avalon/Inria) & J. Bigot (CEA/MdS)
  - Participants: O. Aumage (Storm/Inria) & G. Latu (CEA-IRFM)
  - Weekly audioconference
  - One week visit of Jérôme @ MdS in December 2014



# Conclusion

## Scientific tracks

- Component model evolution
  - Adaptable, static programming CM: HLCM
  - Simple, efficient execution CM: L2C
  - Large scale reconfigurable execution CM: directMod/L2C
  - Efficient component+task CM: Jérôme’s PhD
- HPC applications
  - 3D FFT, domain decomposition, AMR, Gysela5D

## Resources

- C2S@Exa PhD
- 2 year PIA ELCI Post –Doc
- Strong collaboration with CEA (Gysela5D) & Storm (StarPU)