Master 2 - Visualisation Image Performance
University of Orléans (2013-2014)

Implementation and evaluation of 3D FFT parallel algorithms based on software component model

Jérôme RICHARD

October 7th 2014

under the supervision of
Christian PEREZ and Vincent LANORE

University tutor:
Sophie ROBERT

High Performance

- Scientific applications require a lot of computing time
  - Illustris Project
  - Human Brain Project
- Parallelism used to reduce the computation time

High Performance

- Parallel programming paradigms
  - Memory sharing
    - One central memory readable/writable by multiple PEs
    - Synchronization using locks, semaphores, transactional memory, etc.
    - Examples : POSIX Threads, OpenMP, UPC
  - Message passing
    - Messages sent between PEs
    - Synchronization using messages
    - Collective communications
    - Examples : MPI, PVM

High Performance

- Parallel architectures
  - Clusters
  - Supercomputers

- Large variety of hardware parallel architectures used
  - Processing nodes (processor, memory, hard drive, etc.)
  - Accelerator (GPU, FPGA, etc.)
  - Network topology (fat tree, toric 3D grid, hypercube, etc.)

Maximize performances → Adapt the software to the Hardware

High Performance

- Observations
  - Hardware frequently renewed
  - Optimizing for a specific hardware is a time-consuming and costly process

Adapting applications on hardware takes to much time

- Internship
  - 3D FFT computing easy to handle/optimize → variants
  - Uses of software components to handle variability

Adaptation of 3D FFT algorithms using software component

Plan

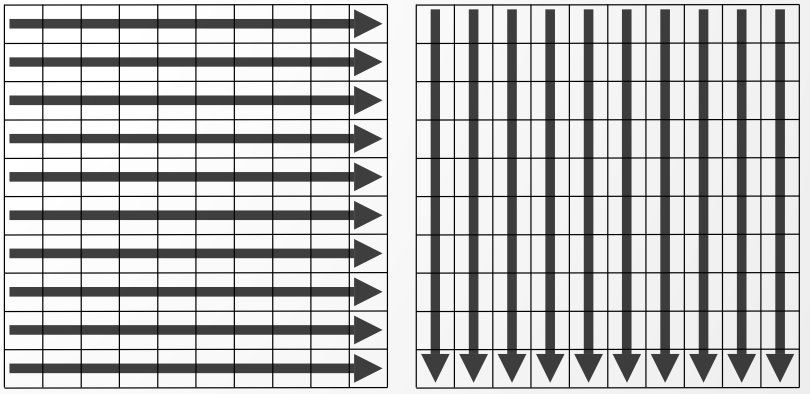
- Context
  - 3D FFT
  - Component Models
- Designing 3D FFT Assemblies with L²C
- Evaluating Re-usability and Performance
- Conclusion and Discussion

Plan

- Context
  - 3D FFT
  - Component Models
- Designing 3D FFT Assemblies with L²C
- Evaluating Re-usability and Performance
- Conclusion and Discussion

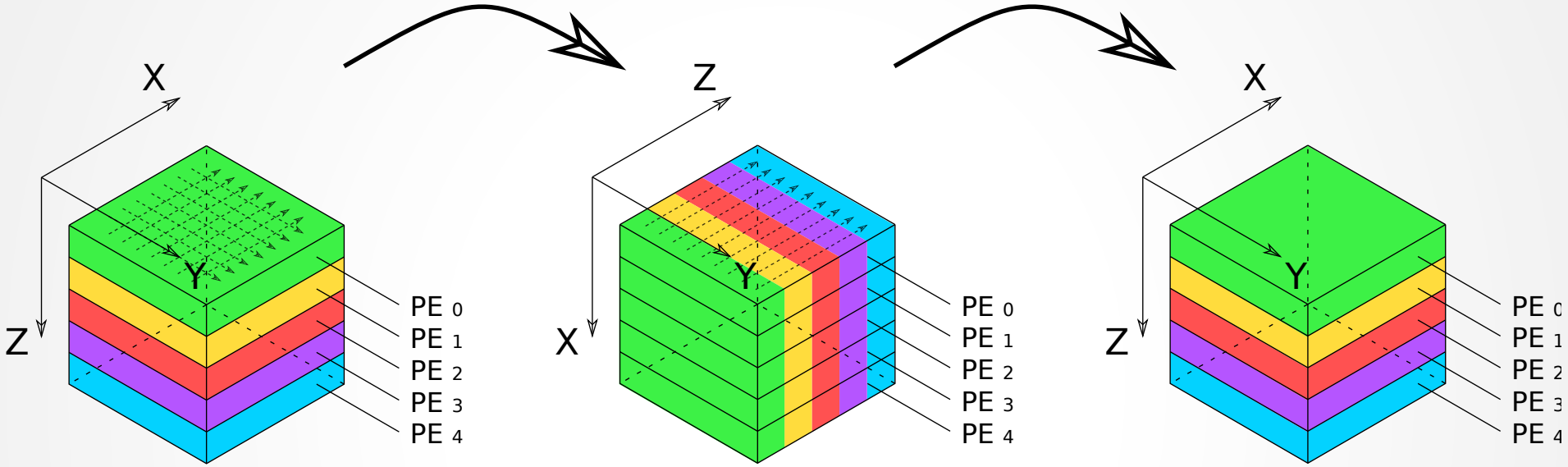
Sequential FFTs

- **F**ast **F**ourier **T**ransform : fast algorithm which converts a discretized signal (matrix) from time domain to frequency domain
- Used in molecular dynamics, astrophysics, meteorology, 3D tomography, seismology, etc. (e.g. GROMACS)
- *Reference algorithm : Cooley-Tukey*
- *Time complexity : O(n log(n))*
- FFT that use >1 dimensions can be decomposed in multiple 1D FFT :



Expensive on larges matrices

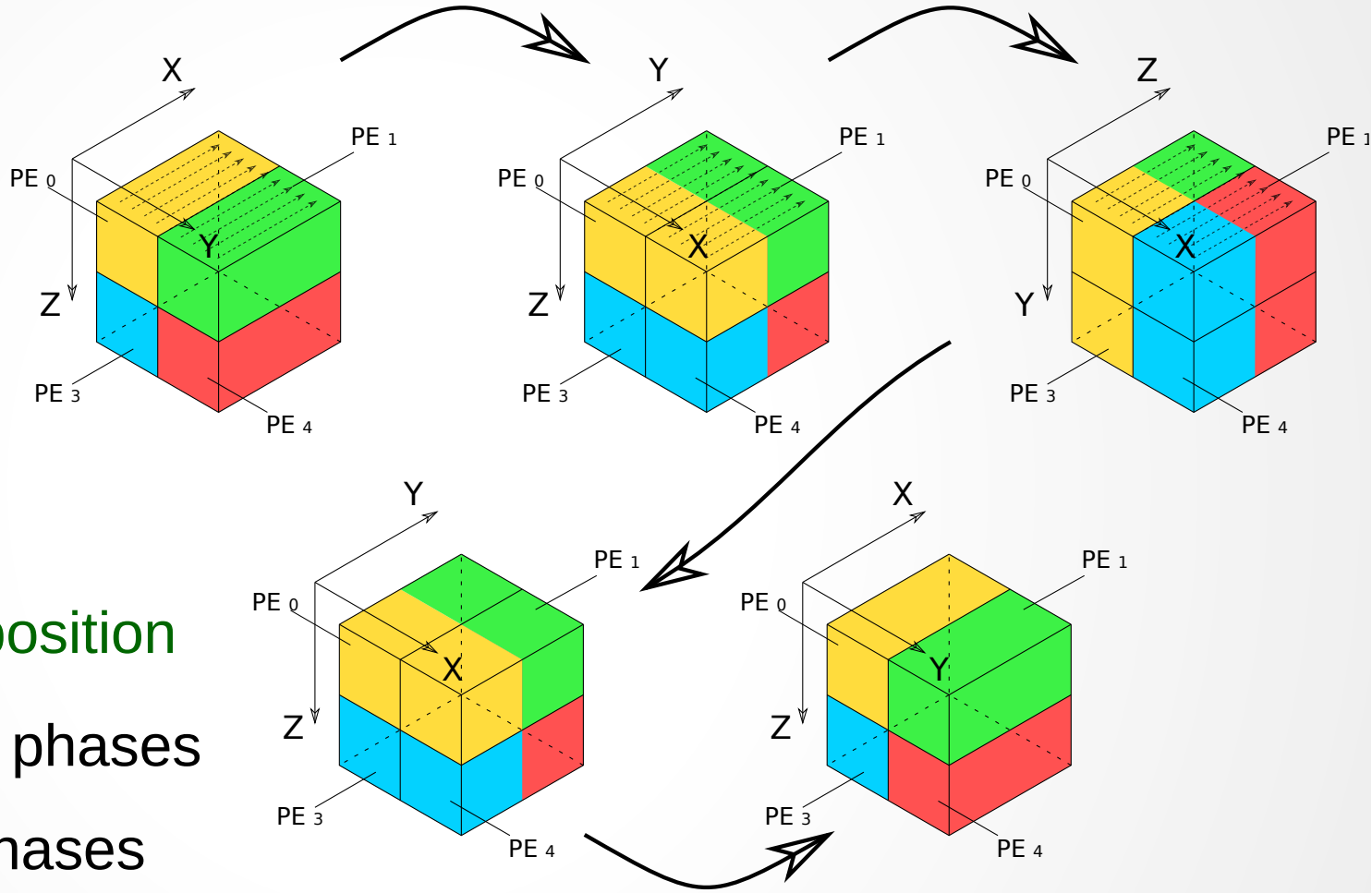
Parallel 3D FFTs



- 1D/Slab decomposition
  - 2 transposition phases
  - 2 calculation phases
  - Scaling limit :
    $Count_{PE} \leq N$ where N is the size of the input matrix

Scaling limit is an issue on current supercomputers

Parallel 3D FFTs

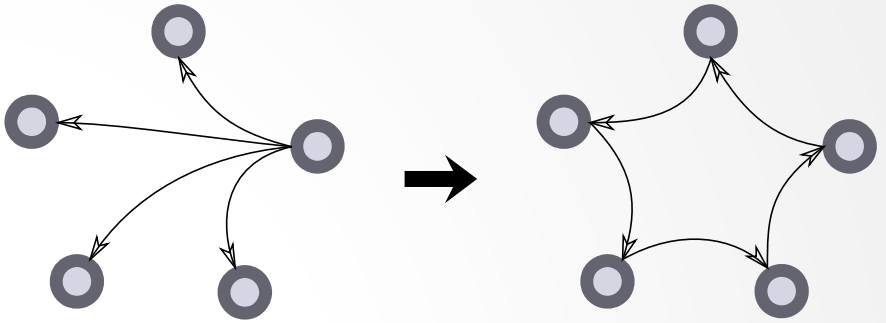


- 2D/Pencil decomposition
  - 4 transposition phases
  - 3 calculation phases
  - Scaling limit :
    $Nombre_{PE} \leq N^2$

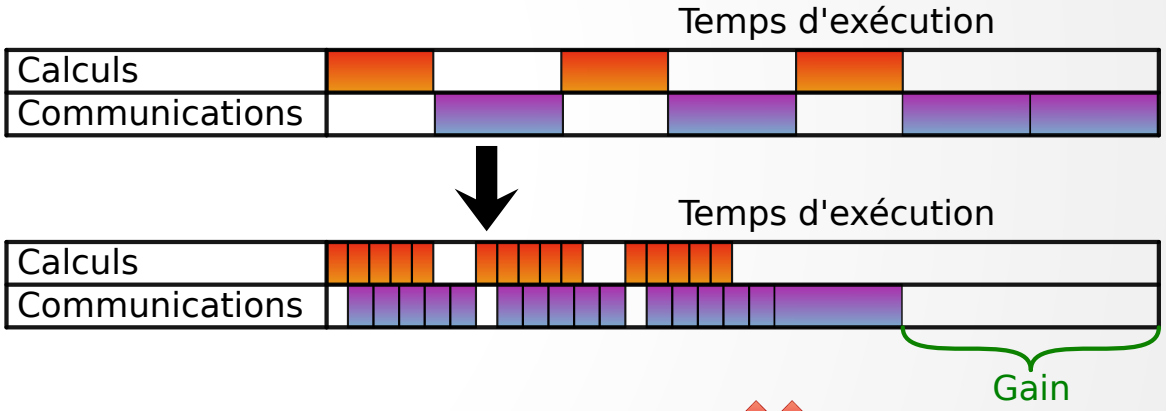
Better scaling limit but more expensive in term of network operation

How to Optimize 3D FFTs

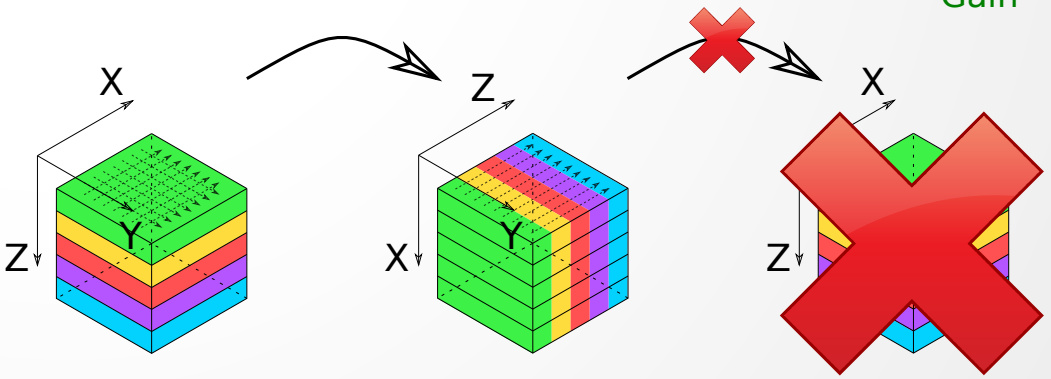
- Adapt the transposition algorithm (total exchange) Bruck1994, Prisacari2013



- Computation-communication overlapping Kandalla2011



- Adapt the number of transpositions Pekurovsky2012



Optimizations depend on hardware and software parameters

Existing Implementations

- Sequential libraries
  - Examples : ESSL, MKL, FFTW, etc.
- Parallel libraries
  - Use sequential libraries
  - Conditional compilation
    - Examples : P3DFFT, 2DECOMP
    - Issues : overlapping of highly optimized small pieces of code
  - Compilation of a high level mathematical description
    - Examples : FFTW, SPIRAL
    - Issues : optimizations integrated into generators/compilers

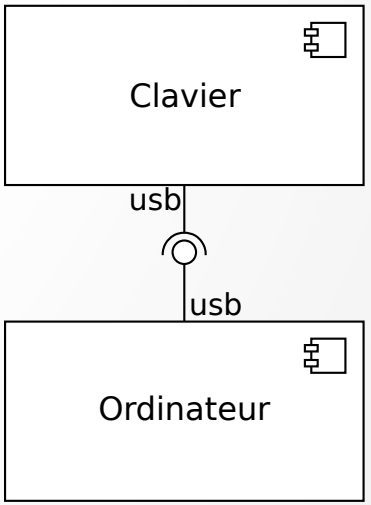
Adding new optimizations is difficult and can be a tricky process

Plan

- Context
  - 3D FFT
  - Component Models
- Designing 3D FFT Assemblies with L²C
- Evaluating Re-usability and Performance
- Conclusion and Discussion

Component Models

- Build an application by assembling component instances
- Component
  - Black box with interfaces
  - Dissociates the interface and its implementation
  - Interacts through its interfaces
- Assembly
  - Component instances
  - Interface connexion
- Benefits
  - Separation of concerns
  - Reuse



Distributed Component Models

- Examples of distributed component models
  - CCM (**C**orba **C**omponent **M**odel)
  - GCM (**G**rid **C**omponent **M**odel)

Overhead not acceptable for HPC applications

- Examples of HPC dedicated component models
  - CCA (**C**ommon **C**omponent **A**rchitecture)

No MPI in the model et granularity limited to process

  - L²C (**L**ow **L**evel **C**omponent)

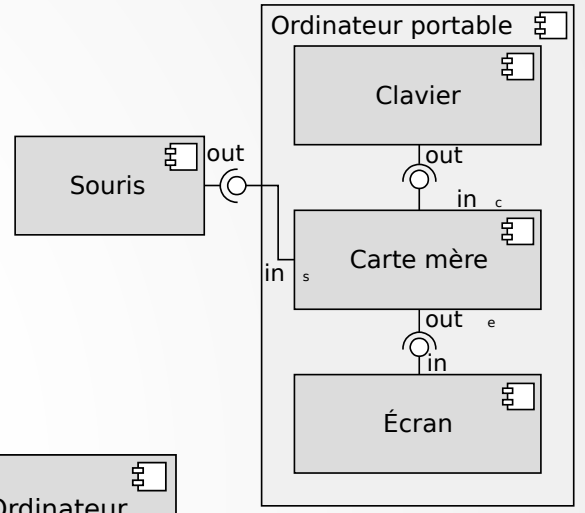
Low **L**evel **C**omponent (L²C)

- Component model that does not hide system issues
- Developed inside the Avalon team
- C++ or FORTRAN components that have attributes and ports
- Intra-process interactions with use/provide ports
  - C++
  - FORTRAN
- Inter-process interactions
  - Message-passing via MPI
  - Remote procedure call via CORBA
- Assemblage describe by a file (processes, attributes, instances...)

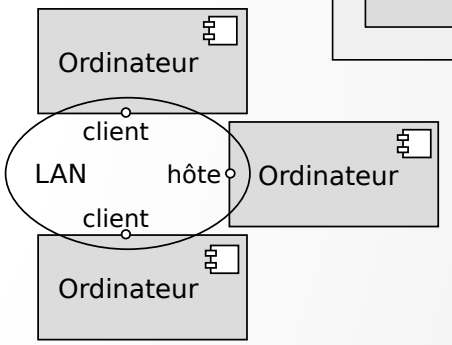
Efficient assembly, but exhaustive description → Generation

Models features

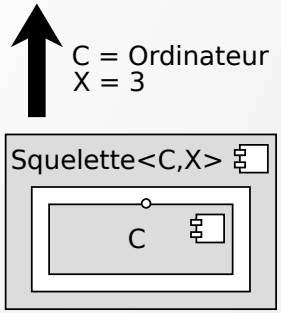
- Hierarchy : component instance can be an assembly (composite components)



- Connectors : element allowing to interconnect multiple instance ports without define specifically how



- Genericity : abstraction of components allowed by using generic component then specialized



HLCM

- **H**igh **L**evel **C**omponent **M**odel
- Developed inside the Avalon team
- Features
  - Hierarchy
  - Generic
  - Based on connectors
- Transform a high level assembly into a low level assembly
- Targeted low level model can be L²C, Gluon++, etc.

Plan

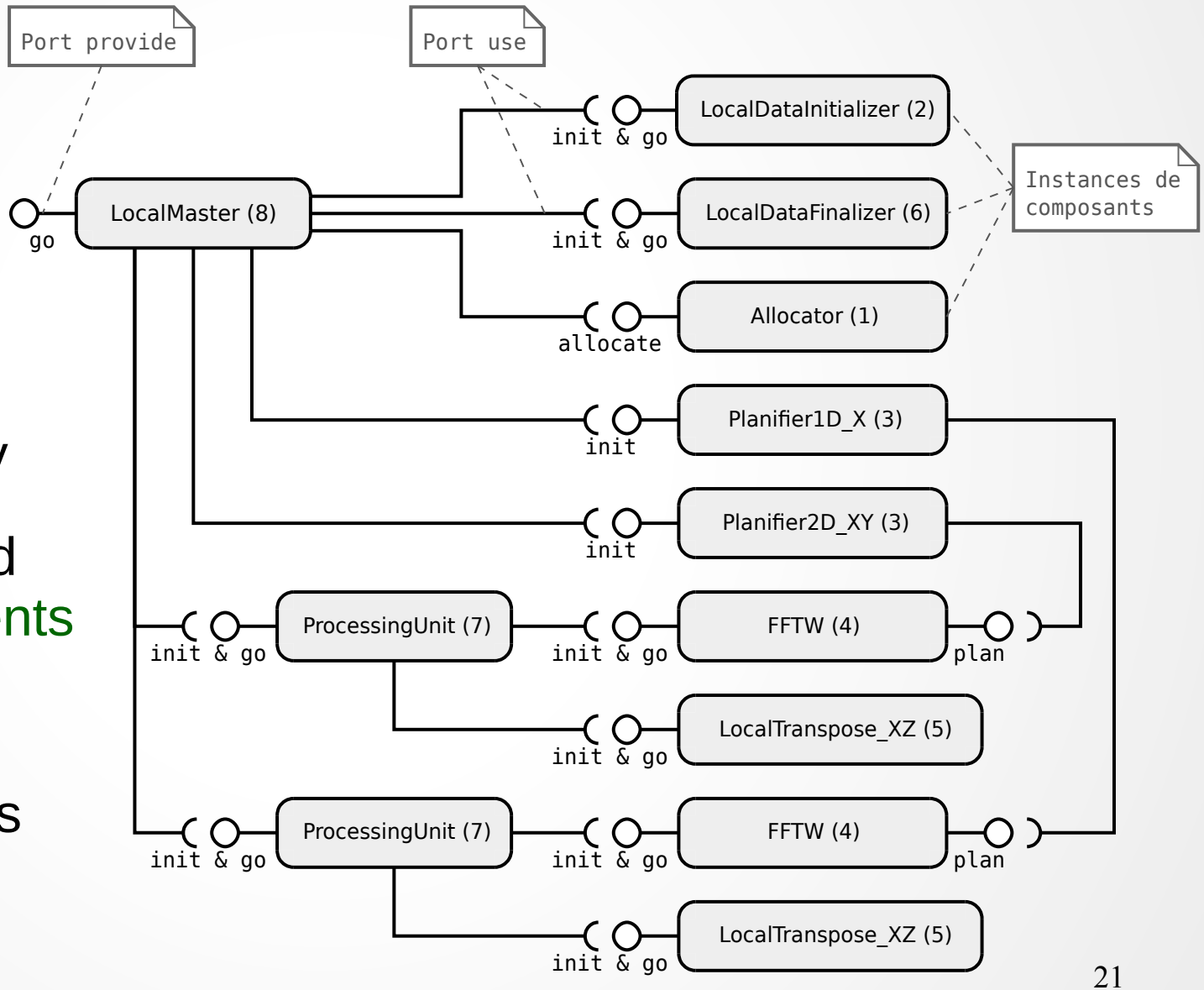
- Context
  - 3D FFT
  - Component Models
- Designing 3D FFT Assemblies with L²C
- Evaluating Re-usability and Performance
- Conclusion and Discussion

FFTs 3D Assembly

- Sequential Assembly
- Distributed Assembly
- Design assembly variants using
  - Local transformations
  - Global transformations

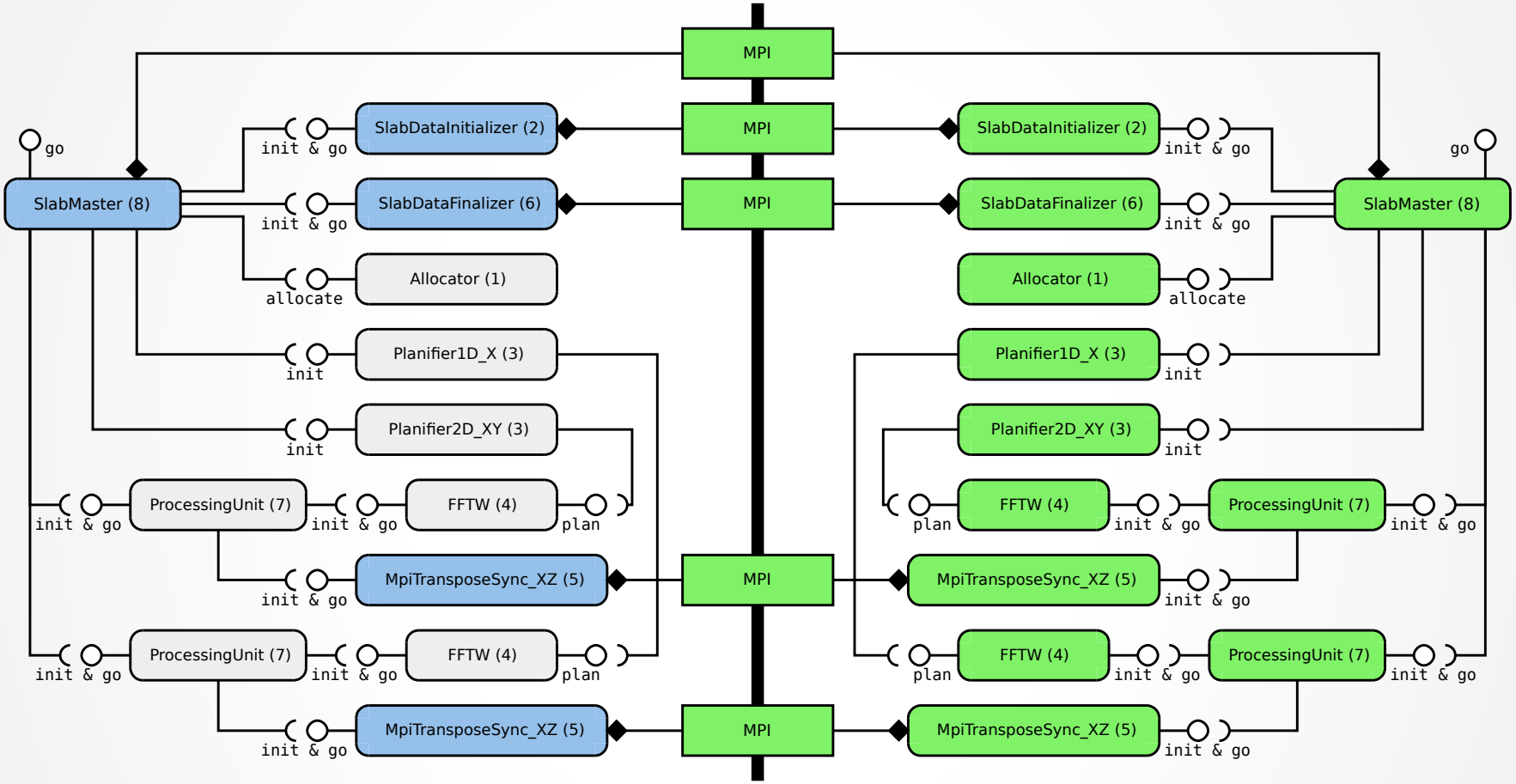
Basic FFT 3D Assembly

- Sequential assembly
- 8 tasks identified and mapped to components
  - 2 control tasks
  - 6 computing tasks



3D FFT sequential algorithms can be implemented in L²C

Distributed FFTs 3D Assembly

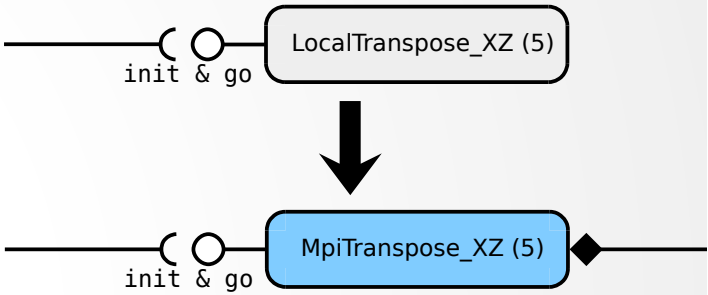


- Assembly duplicated on each MPI process
- 4 components replaced (providing a MPI interface)

1D decomposition can be implemented in L²C

Local Transformations

- Component replacement used to
  - Adapt transpositions
  - Switch the sequential implementation



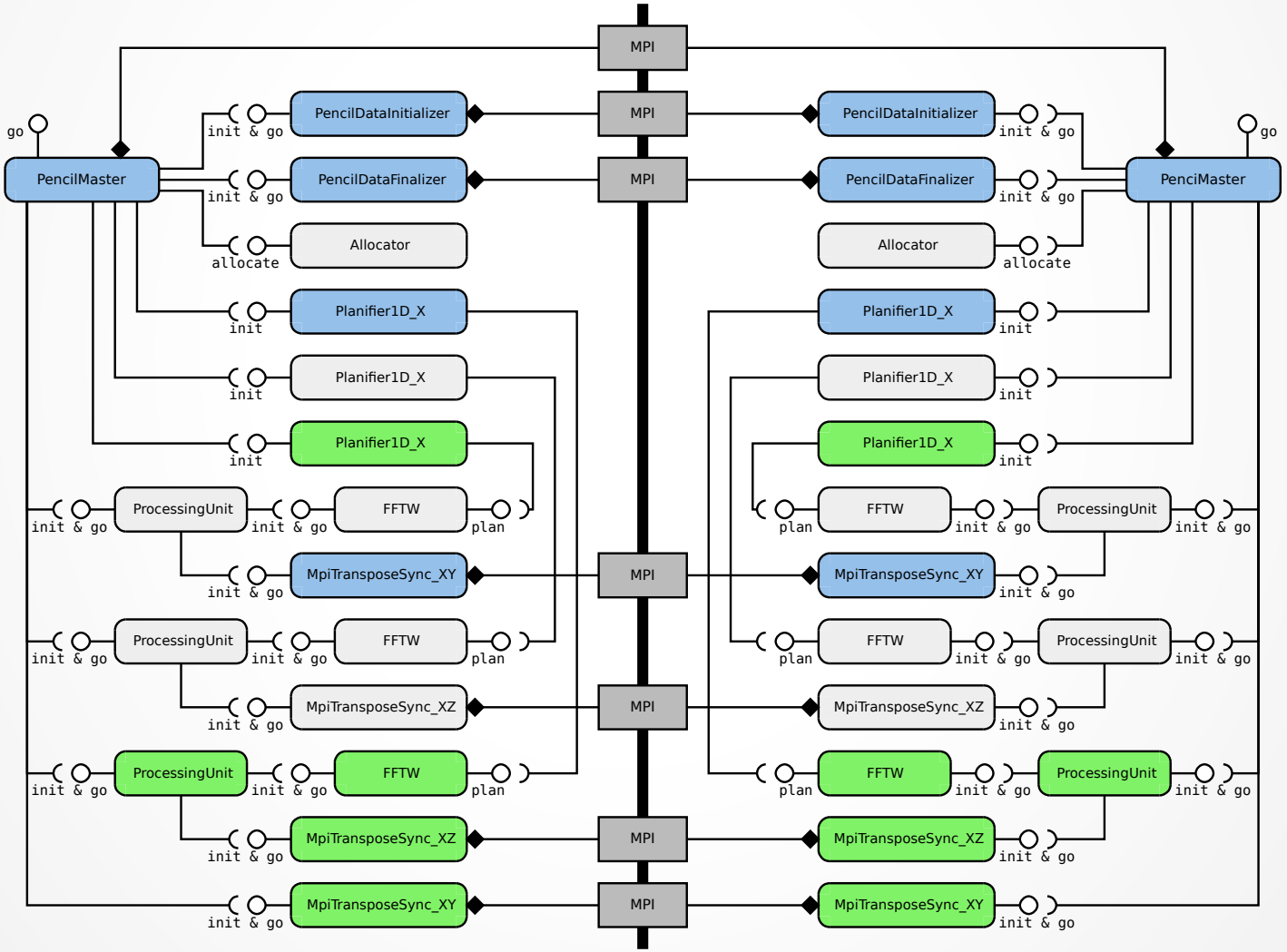
- Attributes configuration used to
  - Load balance work between instances
  - Allowing overlapping

LAD (XML)

```
[…]
<property id="SizeX">
    <value type="uint64">
        256
    </value>
</property>
[…]
```

Global Transformations

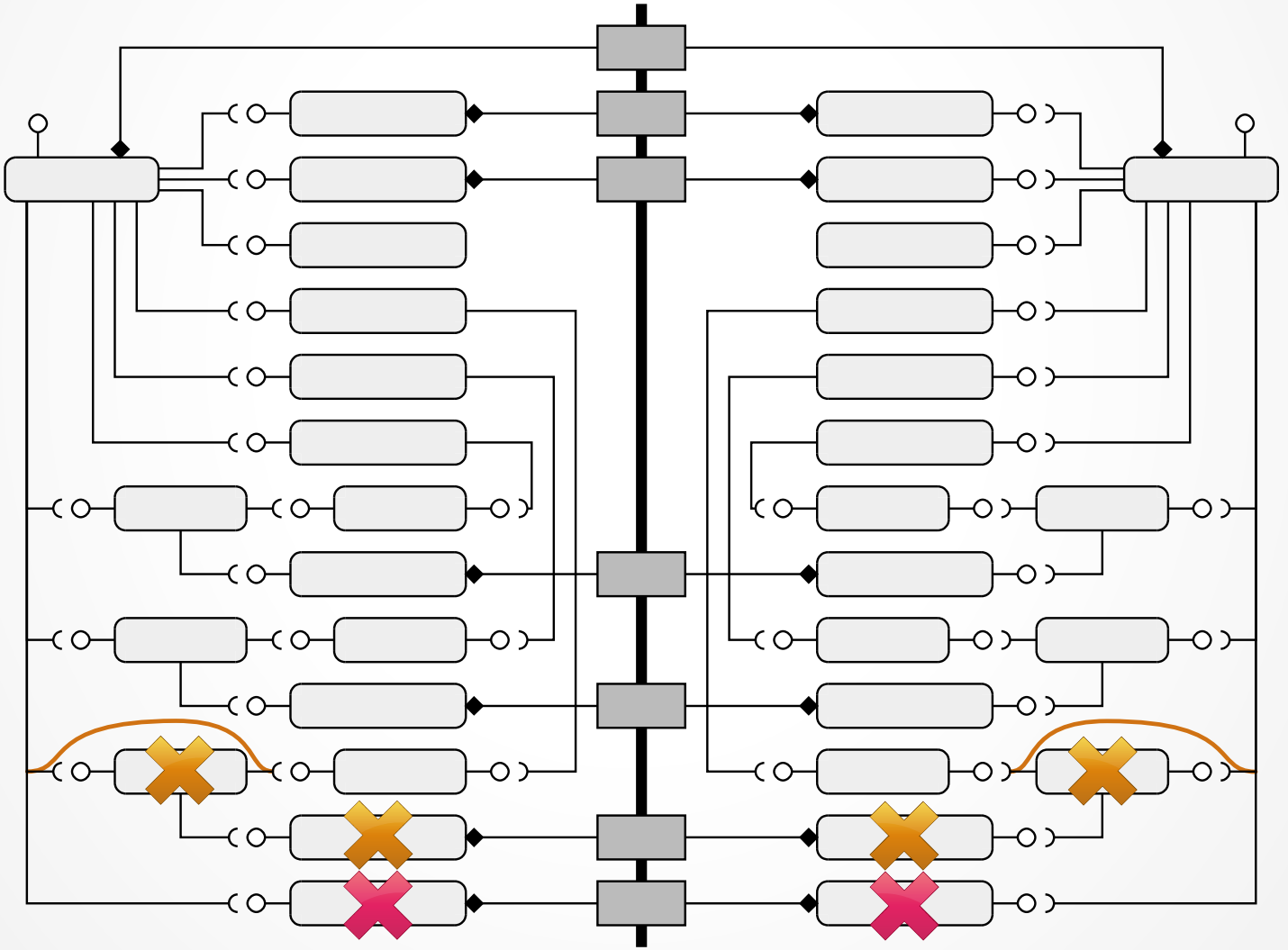
- 2D decomposition



2D decomposition can be implemented in L²C

Global Transformations

- Adaptation of the number of transpositions



Easy to adjust the number of transpositions

Variants

1D decomposition	2D decomposition
L2C_1D_[...]	L2C_2D_[...]

Without load balancing	With load balancing
L2C_XD_[...]	L2C_XDH_[...]

Without overlapping	With overlapping
L2C_XD_[...]	L2C_XDR_[...]

	Transposition count			
	1	2	3	4
1D Decomposition	L2C_1D_1t_xz L2C_1D_1t_yz	L2C_1D_2t_xz L2C_1D_2t_yz	-	-
2D Decomposition	-	L2C_2D_2t_yzx L2C_2D_2t_zxy	L2C_2D_3t_xzy L2C_2D_3t_yxz L2C_2D_3t_zyx	L2C_2D_4t_xyz

- All assembly implemented
  (except those using both overlapping et 2D decomposition)

Some optimization implemented → Many variants

Plan

- Context
  - 3D FFT
  - Component Models
- Designing 3D FFT Assemblies with L²C
- Evaluating Re-usability and Performance
- Conclusion and Discussion

Reuse Analysis

Version	Line count (C++ code)	Proportion of reused code
L2C_1D_2t_xz	927	-
L2C_1D_1t_yz	929	77 %
L2C_1D_2t_yz	929	100 %
L2C_1D_2t_yz_blk	1035	69 %
L2C_1DH_1t_yz	983	80 %
L2C_1DH_2t_yz_blk	1097	72 %
L2C_2D_4t_xyz	1067	87 %
L2C_2D_2t_zxy	1067	100 %
L2C_2DH_2t_zxy	1146	69 %

- Total
  - 30 components
  - 3743 lines of C++ code

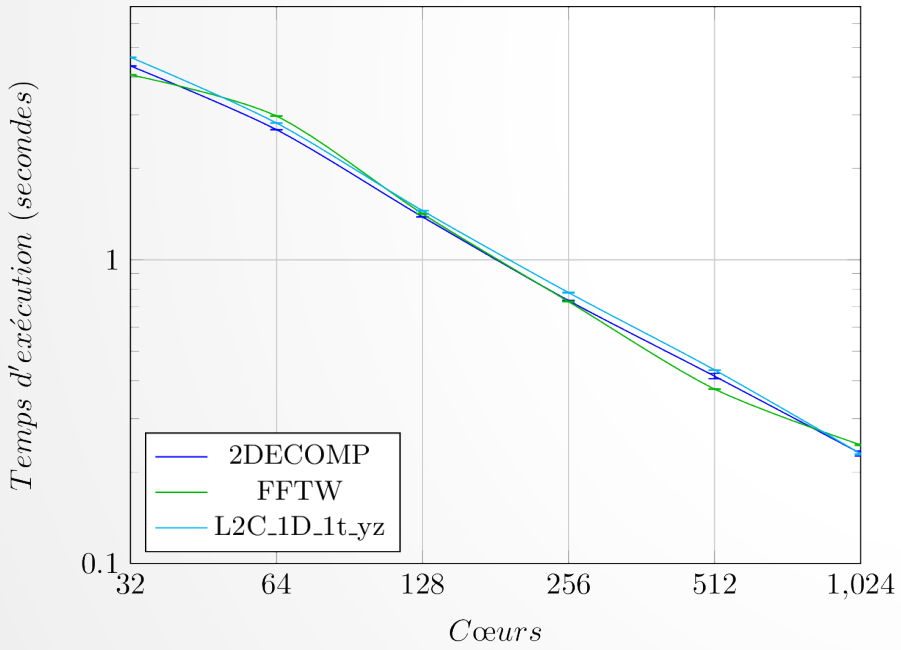
Assembly components are reusable

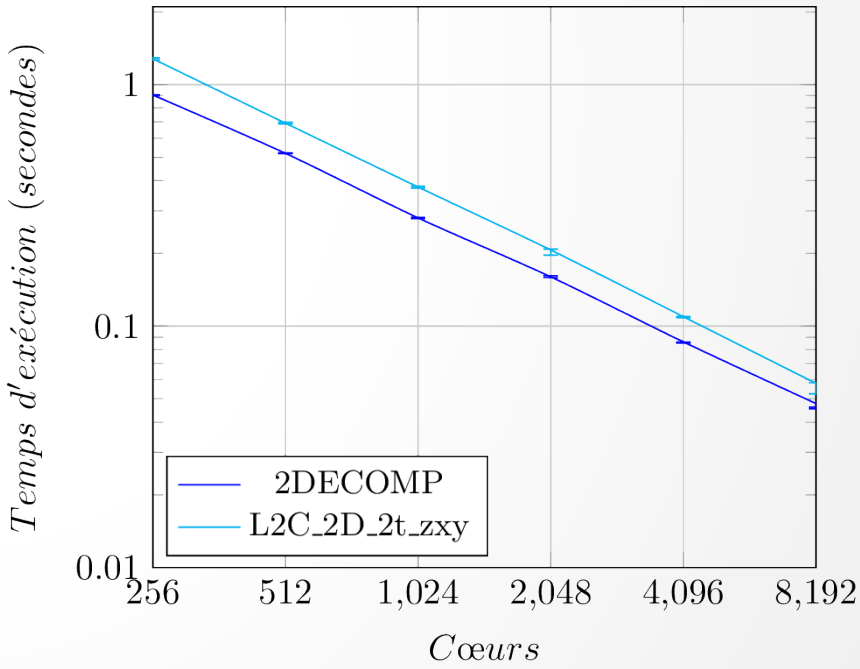
Performance Measurement

Experimental platform	Grid'5000 and Curie
Reference implementation	FFTW and 2DECOMP
Input matrix size	$2^p \times 2^p \times 2^p$ with $p \in \mathbb{N}$
Launch count	100
Result computing method	Average
Error bars	1^{er} et $3^{ème}$ quartiles
Compilers	GCC on Grid'5000 and ICC on Curie
MPI implementation	OpenMPI

Performance Measurement and Scaling

- Large scale experiments on the Curie supercomputer
  - Matrix size : 1024 x 1024 x 1024
  - 1D decomposition
  - 2D decomposition

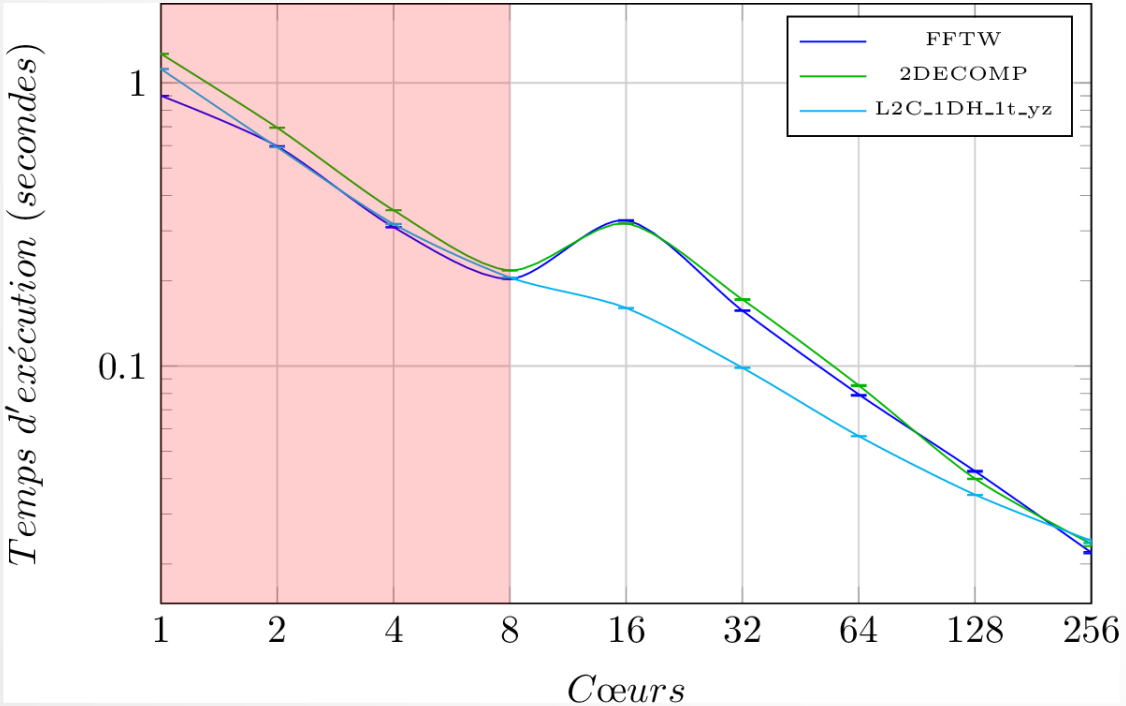


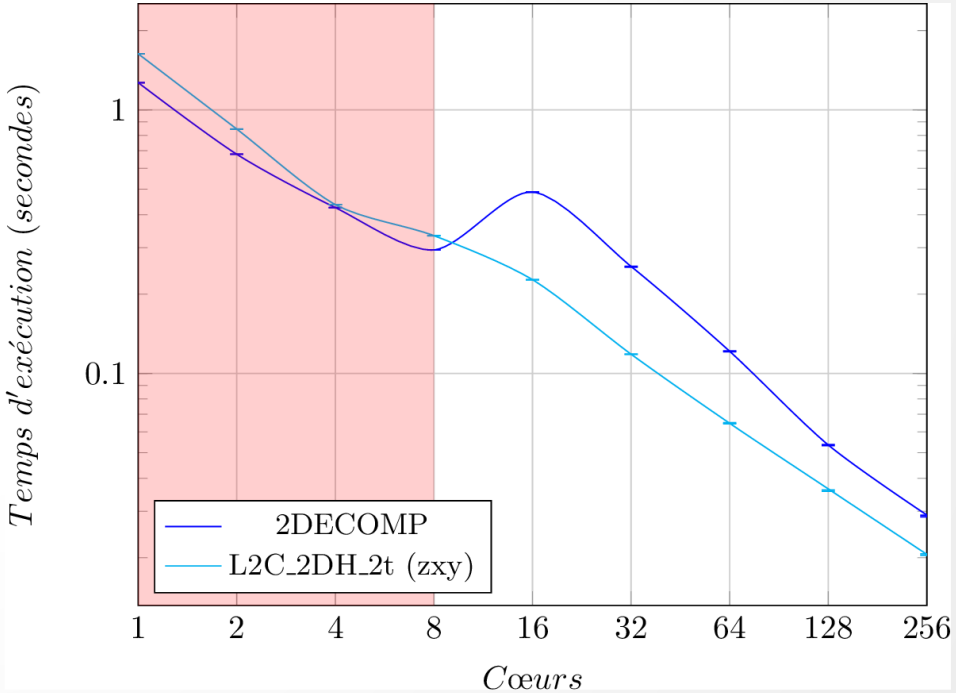


L²C assemblies scale well up to 8192 cores

Variants Evaluation and Analysis

- Heterogeneous experiments (on Grid'5000)
  - Matrix bloc size experimentally adapted
  - 1 “slow” cluster and 1 “fast” cluster used
  - Matrix size : 256 x 256 x 256
  - 1D decomposition
  - 2D decomposition





Assemblies implement load balancing

Plan

- Context
  - 3D FFT
  - Component Models
- Designing 3D FFT Assemblies with L²C
- Evaluating Re-usability and Performance
- Conclusion and Discussion

Conclusion and Perspectives

- Examine whether component models can handle variability of 3D FFT codes
- Contribution
  - Design and implement 3D FFT algorithms with L²C
  - Experiment on Grid'5000 et Curie
- Results
  - Easy to handle variants (code highly reused between assembly)
  - High performance (competitive with reference implementations)
- Perspectives
  - Design higher level assembly using HLCM
  - Implement selection algorithms to automatically specialize assemblies