TRICLADE for CExA

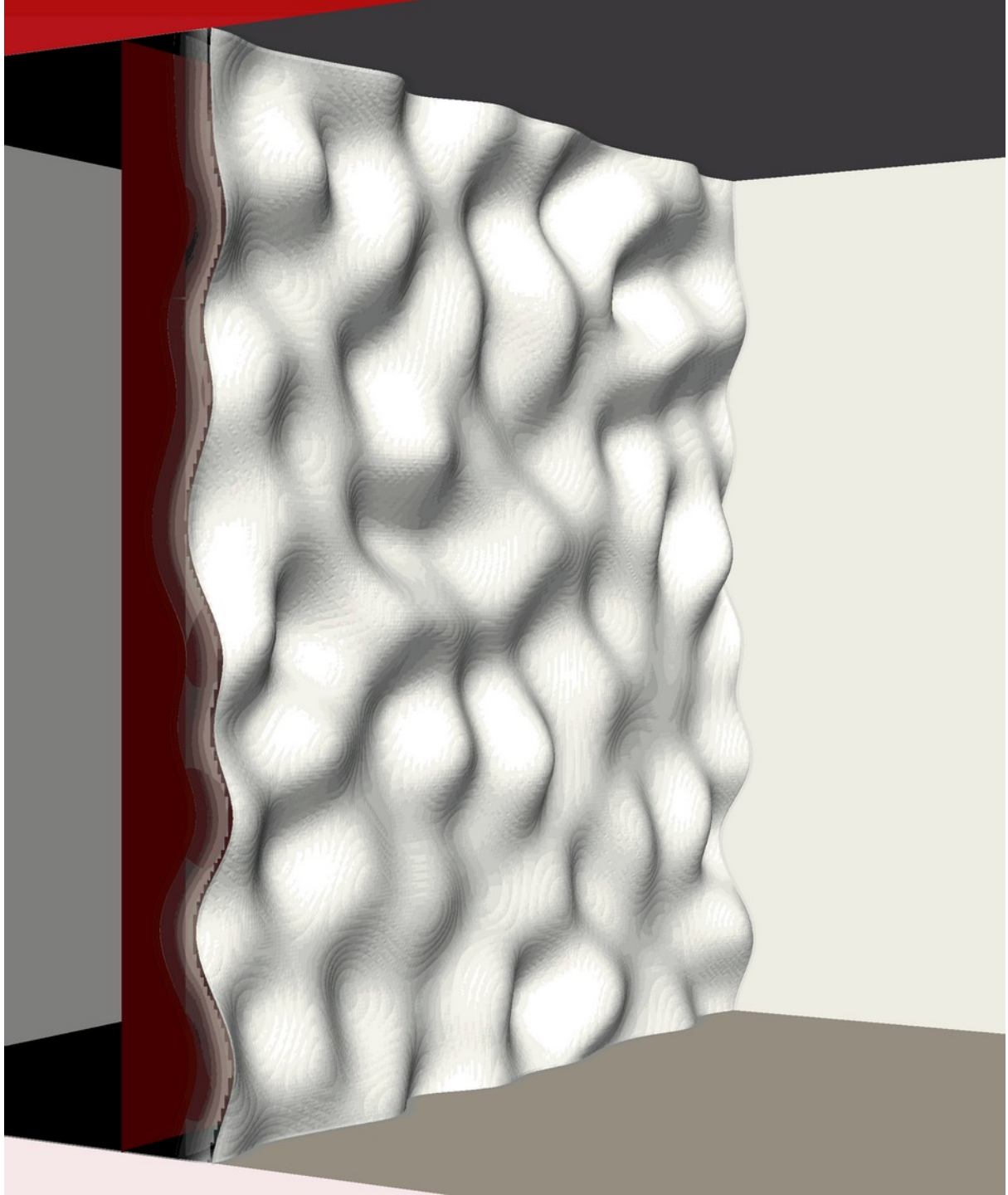
- *Cédric Chevalier on behalf of François Letierce*

- *Comité de suivi de CExA – 2024.10.25*

Some Context

- Study of Turbulent Mixing Zone:
  - Created and developed at fluids interface ;
  - From shock, expansion, acceleration, …
  - Dynamic and structure not fully understood.

- **TRICLADE**:
  - Turbulent binary mixing in a highly compressible environment
  - Navier-Stokes equations
  - Structured Cartesian Mesh
  - « Shock-capturing » numerical schemes

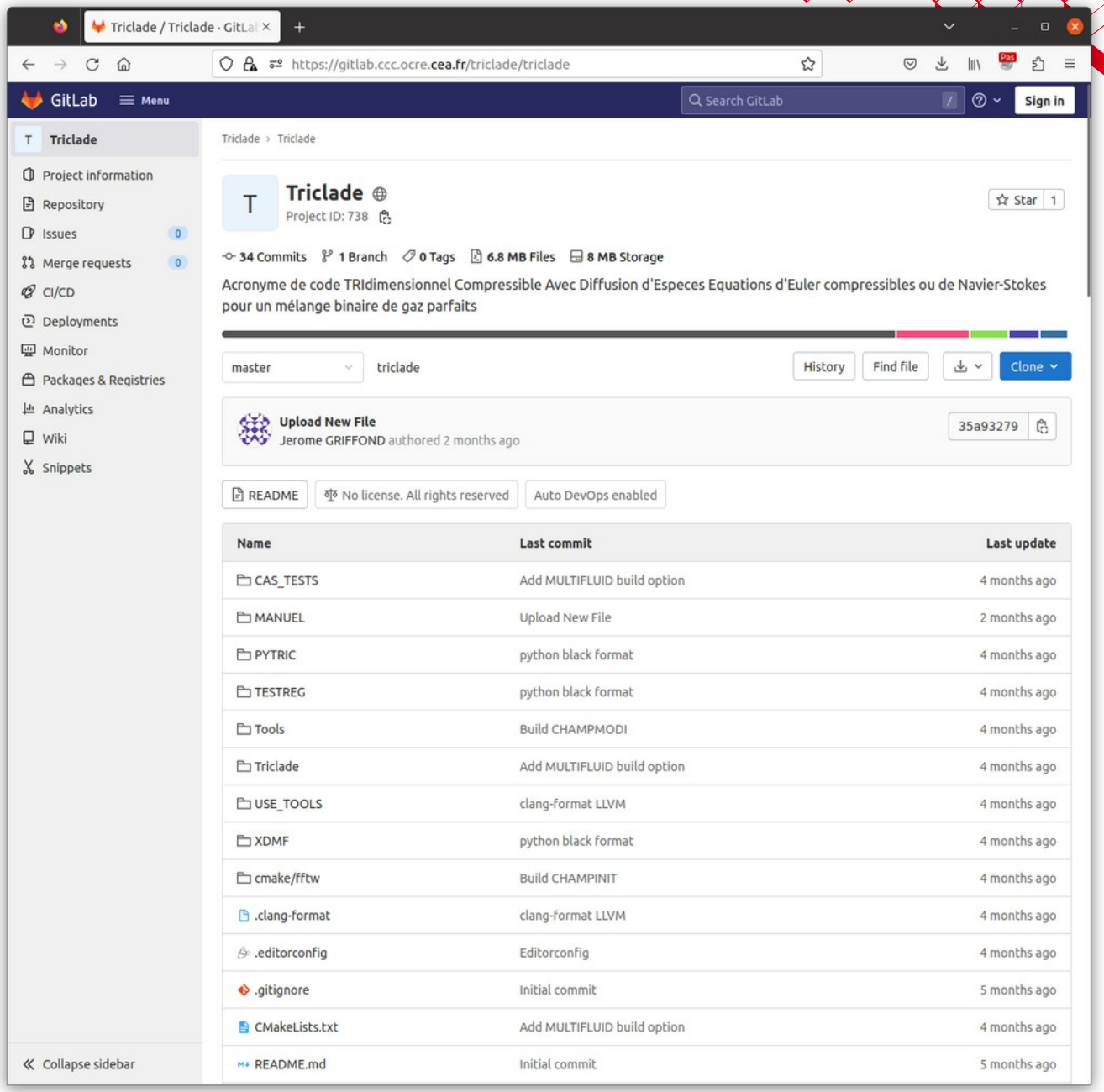


Technical Context

- **Legacy code (almost 30 years old)**
  - ≈ 100 000 lines of C « ++ »
    - Modularity : 1 scheme = 1 module
  - Few external dependencies
    - *MPI*, *FFTW*
  - Quite a lot of internal libraries (IO, initialization, etc.)
    - Proprietary data format…
  - A lot of tooling scripts : configuration, launch
    - Mainly python

First development

- Needs from CEA DAM / CExA
  - *GitLab* CEA « inti »
- Documentation
  - User and developer
- *build system upgrade*
  - *Non standard Makefile => CMake*
- Development and set up of a testing suite
  - Backup reference results



Porting Triclade to GPU

Triclade GPU port was decided

Impacted modules are roughly 10 000 LoC

Regardless of the CExA initiative

Focusing on currently most used features

+ yet to be discovered dependencies...

Migrating to Kokkos

- Replace code « Variables » custom data structure with *Kokkos* ::*View*
  - Several types of variables :
    - Primitive → lifetime : the whole execution
    - Conservative → lifetime : method or class instance
    - Thought of generic management of these different kinds with CExA (DDC?)
  - Collection of variables ?
    - Investigations in progress with CExA

- Rewrite computing loops
  - *Threadsafe* ? If not, deeper refactoring
  - *Parallel_for* + *Lamba*/*Functor*

- Some thoughts to exploit hierarchiecal
  - *Teams* along geometrical axis

Current state

- Design and implementation of test framework
  - Help assessing code validity
  - Use of *LevelDB* (lib *Google*)
  - Backup of reference results
  - Comparisons
    - Using a customisable threshold

```
letiercef@UN00309764:~/Dev/triclade-master/build$ Triclade/tricx ~/Dev/triclade-master/TESTREG/CAS_TESTS/sod2d_fl --createRef
 LANCEMENT
              ************************************************
              *                                              *
              *              T R I C L A D E S  - 3 D        *
              *              (c) CEA/DAM  -  2021     V4.8   *
              ************************************************


Manuel utilisateur accessible dans :
/home/letiercef/Dev/triclade-master/Triclade/MANUEL/manuel.pdf

[LevelDB] Create a reference for /home/letiercef/Dev/triclade-master/TESTREG/CAS_TESTS/sod2d_fl while running the simulation.
        #              *----------------
        #              *--Def du dom ----
```

- No abstraction over Kokkos
  - From other Kokkos migrations

```
FIN NORMALE DU CAS sur ARRET CONTROLE (Nombre iterations maxi atteint) avec 1 PE(s) en 1.50515 secs
Gtime  6.04152e-06 en 1.44997

Documentation doxygen :
/home/letiercef/Dev/triclade-master/Triclade/Triclade/DOXYDOC/index.html

Manuel utilisateur :
/home/letiercef/Dev/triclade-master/Triclade/MANUEL/manuel.pdf
[LevelDB] Comparing current simulation to reference ...
 => OK ! Results are similar.
```

- Simple changes for code « init »
  - New options for the user
  - New method overloads for Kokkos::View

```
using TargetExec = Kokkos::DefaultExecutionSpace;

using TargetMem = TargetExec::memory_space;

using KokkosViewReel2D = Kokkos::View<Reel**, TargetMem>;

using KokkosViewReel3D = Kokkos::View<Reel***, TargetMem>;
```

Some technical issues

- Handling both legacy and Kokkos' variables
  - Types *POD*(*****)
    - Explicitly passed as parameters for functions/methods
  - Memory allocation :
    - In scattered functions
    - Called everywhere: from main, inside objects or functions
  - Use of generic abstract methods everywhere : hard to modify

```
template <class TypeVar, class TypeN>
inline void CreeTab(TypeVar ****&var, TypeN n, TypeN n1, TypeN n2, TypeN n3)
```

```
class FluxM5lmHLLC2 : public FluxM5lm {
  typedef FluxM5lm super;
```

```
public:
  //! calcul du Flux en 2D
  /*!
    \param info objet contenant les informations specifiques du processeur
    \param flux pointeur du tableau contenant les flux
    \param varL pointeur des valeurs a gauche de la face
    \param varR pointeur des valeurs a droite de la face
    \param idir direction du flux
    \param lgn indice de ligne
  */
  virtual void Flux(InfosProc &info, Reel **flux, Reel **varL, Reel **varR,
                    int idir, int lgn) = 0;
```

Some technical issues

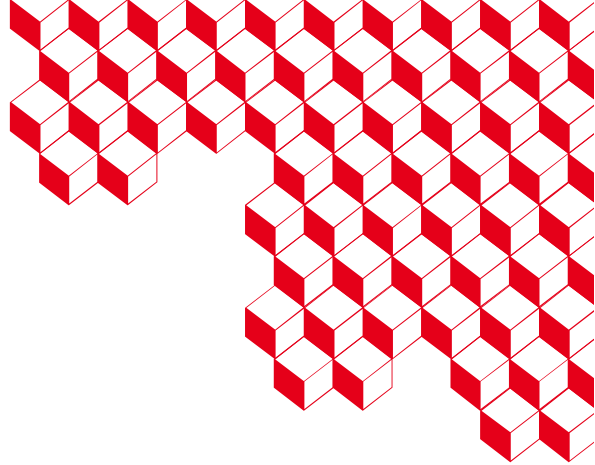
- Considerably more refactoring than expected
  - *Hardcoded data type* → the subset to port cannot be trivially isolated (From 10K lines impacted to 100K ...)
    - *Fork* + deletion of other code paths
    - Massive string replacement (*sed* + *regex* + AI + prayers)
    - Setup a small abstraction mechanism + interface update

- Incremental changes strategy
  - Expensive
  - Intermediate numerical validations at each steps
  - Use of a small proxy for data: [i][j][k] ↔ (i,j,k)
    - Brak project from CExA

Conclusion

- GPU port of Triclade has begun (but with some delays due to the necessary pre-port refactoring)
- Goal is still a first Grace-Hopper (Exa1 HE) run with some GPU offloads by the end of the year

- The preliminary overview has underestimated some difficulties (interdependent):
  - The lack of common abstraction for data types (nested raw pointers)
  - Generic abstract calls for module methods making module tightly coupled
  - (The lack of a proper testing framework)
  - (The technical debt of an old legacy code,+*2500 compilation warnings with modern compilers, Bugs, Memory Leaks, no multi-threading*)
- These difficulties are not related to Kokkos : adapting an existing code as some drawbacks comparing to write a new one
- Being close to CExA has considerably helped to
  - Design and choose technical designs (by exploiting the shared community knowledge)
  - Ease development by developing some helper tools
- Other CExA developments are of interest and will be used later
  - Kokkos-COMM, Kokkos-FFT, ...



Thank you