A Case Study on the Impact of Chapel within an Academic Computational Aerodynamic Laboratory

Éric Laurendeau[1]

1 *Polytechnique Montreal, Quebec, H3T 1J4, Canada*

NSERC CRSNG

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

Table of contents

1. Context
2. CHAMPS
3. Scalability
4. Numerical Verification
5. Conclusion

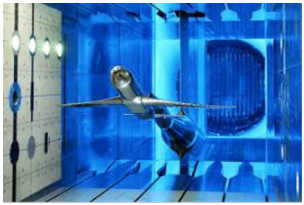
Aerospace Engineering

- National Objectives (Commercial & Defence)
- Flight testing extremely costly (Dollars and Humans)
- Simulation-Based Engineering have emerged from R&D ('90s-2010') to production
  - Disciplinary (manufacturing, aerodynamics, etc.)
- Technology push:
  - Multidisciplinary : link 'fields'
  - fully coupled systems : link 'software'
- Solution: democratizing HPC
  - Unified OS, languages, memory, architecture

Aerodynamic Design Toolset

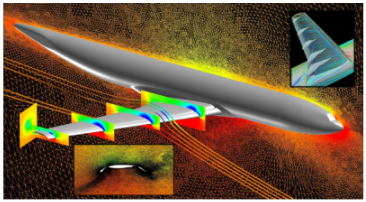
Bombardier

Flight tests~100 000$/hr



ETW

Wind-tunnel ~1000-10 000$/hr



NASA

Numerical Simulations (CFD)~100-1000$/hr

8

Workflow

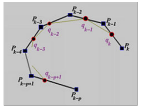
physics

$$\rho \underbrace{\left(\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u}\right)}_{\text{Acceleration}} = \underbrace{-\nabla p}_{\text{Pressure}} + \underbrace{\nu \triangle \vec{u}}_{\text{Viscosity}}$$

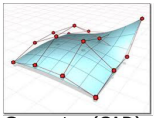
$$\underbrace{\nabla \cdot \vec{u} = 0}_{\text{Continuity Equation}}$$

MIT

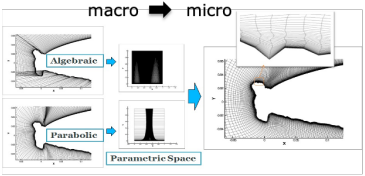
mathematics



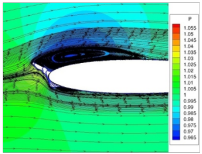
Parametrisation (NURBS)



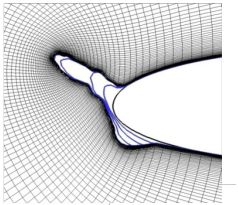
Geometry (CAD)



Mesh generation
0.01-10 billions unknowns



solution:
i) flow, ii) droplets



Time step

Proprietary source codes

Supercomputers

University-based Education Workflow towards CFD knowledge

- Technology used to be in academia
  - Research centers
    - Industry
- Handled by Professors, Research Associates, Post-Docs, highly specialized skill sets
- Now handled by M.Sc. students
  - Push towards undergraduate training
- Example 1: Polytechnique Montréal adopted Python in all U. Grad courses (no more Matlab)
  - Python introduced at Bombardier through students! Technology push
- Example 2: Chapel used in 3D Navier-Stokes solver, will it see same success? (hopefully!)

University-based Educational Challenges

- Physics
- Applied Mathematics
- Numerical analysis
- Programming languages
- OPEN-MP (M.Sc.)
- MPI (Ph.D., slow progress)
- Mixed CPU-GPU (failure in technology push)
- Time-to-debug (1 Million+ lines, 10hrs runs)

How do you fit that:

- in U. Grad?
- in Ph.D. (with M.SC. dropping?)

So far, experience has shown Chapel is step change towards this goal.

Chapel for CFD

Productivity

- The research field of CFD evolves rapidly and is competitive
- Requires quick implementation of complex algorithms over distributed memory

Fast

- The inherent computational cost demands fast software

Portable and Scalable

- 2D cases on a desktop
- Large-scale 3D cases over 500+ cores
- 1 code portable to any hardware

CHapel Multi-Physics Simulation (CHAMPS)

In 2.5 years:

- 3D Unstructured Reynolds Average Navier-Stokes flow solver (supports 1D, 2D and 3D grids)
- Convective flux schemes: Roe, AUSM and MATD (Second order finite volume)
- SA, SST-V and Langtry-Menter transitional turbulence models (with variants)
- Explicit solver (Runge Kutta) and implicit solvers (SGS, GMRES)
- Jacobian-Free Newton-Krylov
- Global Stability Analysis
- Interface with external libraries: MKL, CGNS, METIS and PETSC
- Icing modules : droplet trajetories, thermodynamic exchanges on surface, surface deformation, volume remeshing/mesh deformation, stochastic ice accretion
- Aeroelastic module : structural simulation, volume mesh deformation

CHapel Multi-Physics Simulation (CHAMPS)

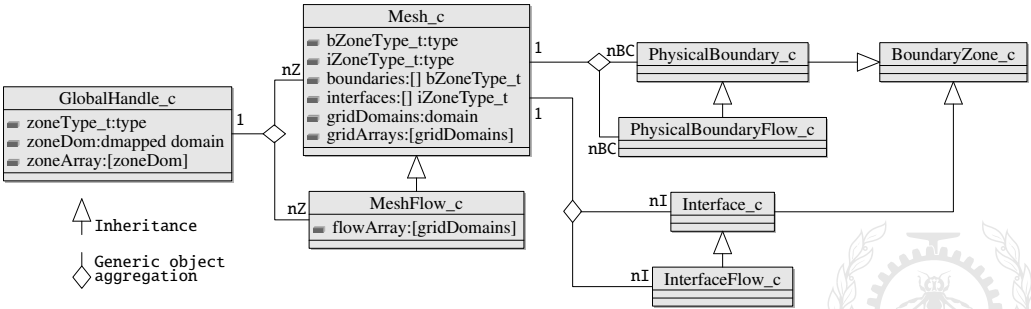
CHAMPS is now used for:

- Fifth AIAA Drag Prediction Workshop;
- First AIAA Ice Prediction Workshop;
- Fourth AIAA High Lift Prediction Workshop;
- Upcoming Seventh AIAA Drag Prediction Workshop.

These workshops show CHAMPS is mature enough to be tested on industrial level applications

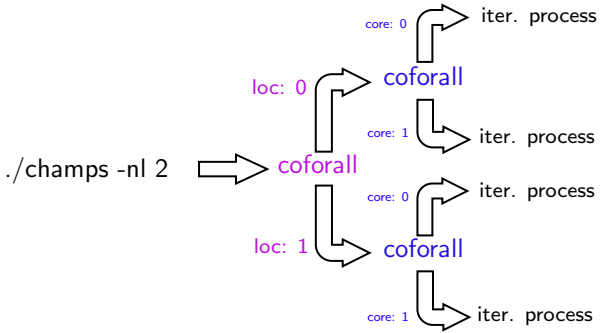
CHAMPS

- A Multi-physics problem requires different computational grids
- Type aliases are used to define these various computational domains
- In the example below, the droplet components could be added as inheriting from the flow components (since they run on the same volume mesh), while other kind of components (thermodynamic, geometry, etc.) would directly inherit from *Mesh_c*



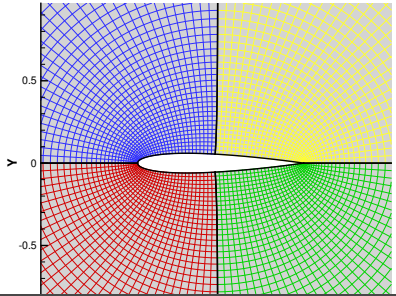
Parallelism over distributed memory

- Single execution
- A task is created for every available *locale*
- The task on a *locale* creates a task for every mesh zone on that *locale* after which the iteration process is launched



Communication

```
1  proc performInterfaceExchanges(zone, exchangeType)
2  {
3      // Fill buffers
4      local do zone.prepareExchange(exchangeType);
5      allLocalesBarrier.barrier();
6      // Read buffers
7      zone.exchangeInterfaces(exchangeType);
8      allLocalesBarrier.barrier();
9  }
```



Communication (Cont'd)

```
1  class ICScalar_c :
       InterfaceConnect_c
2  {
3  type type_t = real;
4  var donorBuffer_ : [D] type_t;
5  var recvBuffer_ : [D] type_t;
6
7  proc fillBuffer(zone,bZone) {
8  //Fill buffer according to class
9  }
10 proc exchange(zone,bZone){
11 // Remote copy buffer from shared
12 // reference (assigned previously
       )
13 recvBuffer_ = bZone.
       remoteIcScalar_.donorBuffer_;
14 // Read buffer according to class
15 }
16 }
```

```
1  class InterfaceZone_c :
       BoundaryZone_c
2  {
3  var icScalar_:shared ICScalar_c;
4  var remoteIcScalar_:shared
       ICScalar_c;
5  }
```

Connection through shared objects

Interface zones contain a reference to a local instance of the *InterfaceConnect_c* object that contains local buffers and a reference to its remote counterpart through the **shared** memory management

Programming productivity

CHAMPS solves non linear PDE equations of the form

$$\Omega \frac{\partial \boldsymbol{W}}{\partial t} = -\boldsymbol{R}(\boldsymbol{W}) \tag{1}$$

Here, $\boldsymbol{W}$ is the solution and $\boldsymbol{R}(\boldsymbol{W})$ is the summation of the:

- Fluxes, computed on the faces of the control volumes (cells);
- Source terms, computed on the center of the control volumes.

Usually we are looking at a steady-state solution, but we are iterating in time up to a solution $\frac{\partial W}{\partial t} = R(W) = 0$. This involves linearizing the equations and solving systems of the form:

$$\boldsymbol{A}\Delta\boldsymbol{W} = -\boldsymbol{R}(\boldsymbol{W}) \tag{2}$$

Programming productivity

Multiphysics applications involve multiple PDE's

- Eikonal equation for the wall distance;
- Navier-Stokes equations (5 coupled equations);
- Turbulence models (1 to 4 equations);
- Droplets equations (4 coupled equations)

For productivity it is preferable to re-use functions between modules. For this the object oriented features of Chapel are used.
This allowed to:

- Define general structures for the LHS and RHS (Left- and Right-Hand sides of the equations);
- Define general structures for fluxes and source terms computation (to fill LHS and RHS);
- Define general structures for updating a solution;
- Program general linear solver for this structure (Runge-Kutta, Gauss-Seigel, GMRES, etc.) to solve any $AX = B$ system of equations;

Programming productivity

Example 1: Summer student was able to implement and verify a turbulence model (2 equations $k - \omega$) in a 4 months internship by:

- Create a class of turbulence model type (based on 1 equation Spalart-Allmaras);
- Implement the equations for fluxes, source terms and jacobian;
- All other functions are re-used.

Example 2: We used to have a Gauss-Seidel (SGS) linear solver. Implementation of a GMRES solver was:

- First done for the flow solver.
- Readily available for all PDEs.

Example 3: We used to solve turbulence model and flow in a segregated manner. A coupled version was done simply by assembling a global RHS and LHS and re-using functions from existing models.

Programming productivity

All this could have been done in C or C++ but...

No strong attachment to another language

- Needed to redo our previous C code (2D structured on shared memory without code reuse);
- New students (Master especially) need to learn the language(s) of the code during their years of study (2 for Master students);

Constraints with C

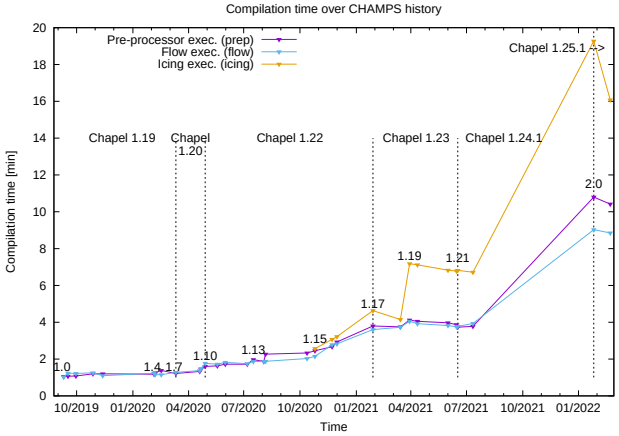
- Not object oriented, so the inheritance and generic style would have to be mimicked, which would be difficult to apply without errors for beginners;
- MPI would need to be added in the list of knowledge to acquire.

Constraints with C++

- Not the language best known to the creators of CHAMPS;
- Oriented object programming, but can be tedious with code duplication in headers (prone to mistakes for beginners), which decreases productivity;
- MPI would need to be added in the list of knowledge to acquire.

Challenges of the Chapel Implementation

One of the main drawbacks of the growth of CHAMPS with new components or modules is the increase in compilation time and required memory, especially for the icing executable. At its highest point, the compilation time could take around **20 minutes**, whereas memory usage was seen to reach up to **35GB of memory** (RAM).



Challenges of the Chapel implementation

Why?

Simple causes (introduced by unfamiliar students with the Chapel language):

- The overuse of generic function arguments in some modules;
- The `use` of too many modules everywhere even when they were not required.

Complex causes :

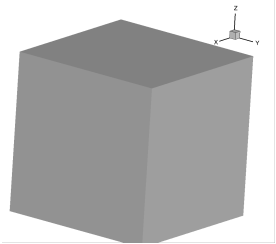
- The addition of new components in the code (new modules);
- The addition of new variations for models, such as new schemes for the fluxes or turbulence model variants;
- The duplication of generic functions for multiple flavors of the mesh and model objects (even outside programmed combinations).

How to reduce the compilation costs?

- Split the compilation in two phases to reduce the peak memory usage:
  1. the generation of the C code from the Chapel files;
  2. the compilation of the C code.
- Address the overuse of generic functions arguments and modules;
- Properly use `where` statements to limit the duplication of generic functions.

Scalability

- The scalability is evaluated on a Cartesian grid
- The cube has farfield boundary conditions only



Strong Scaling

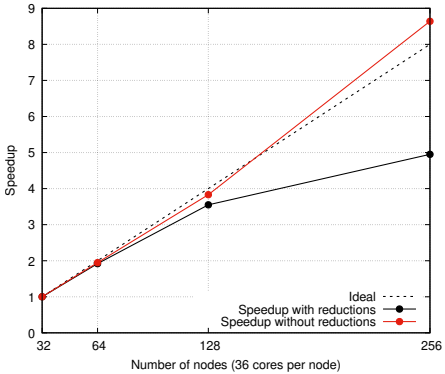
The cube is discretized with 800 elements in every direction (i,j,k) for a total of 512M elements

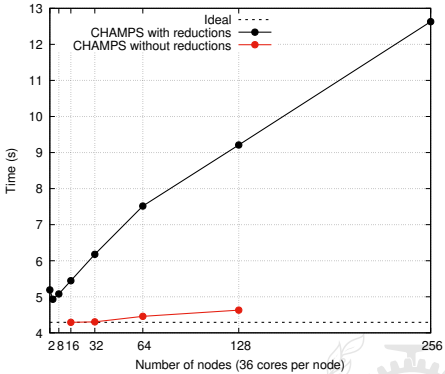
Weak Scaling

The problem size ($\sim$ 1M per *locale*) is scaled with the number of *locales*

Scalability

- The scaling result is highly impacted by global reductions (i.e. lift, drag and residual values to monitor flow convergence)
- Linear scalability is maintained at 9216 cores without these reductions



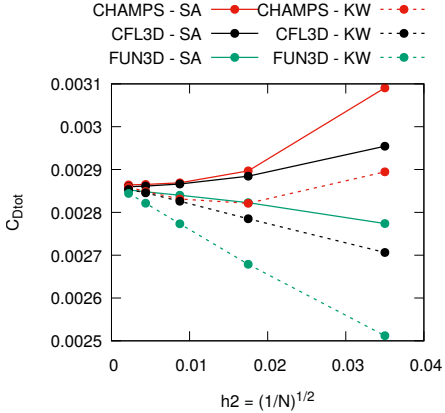


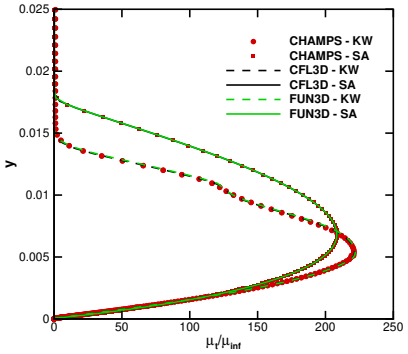
(Left) - Strong Scaling (Right) - Weak Scaling

Numerical Verification

Flat Plate Turbulence Model Verification

- SA and KW models are verified against CFL3D and FUN3D
- Similar grid convergence is achieved for C_D

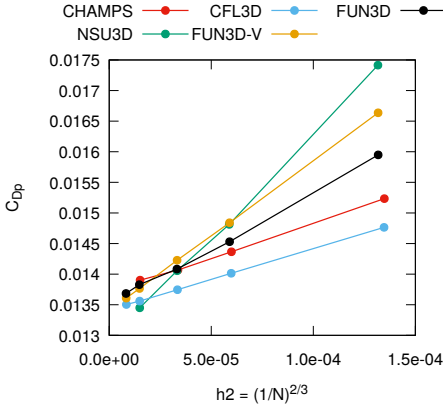


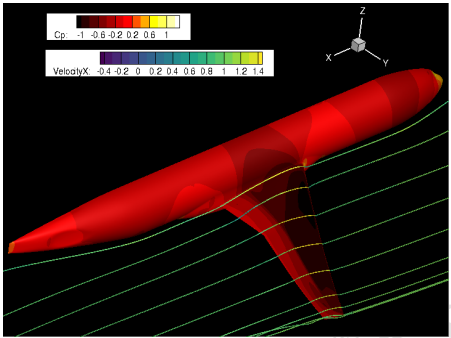


Numerical Verification

Fifth Drag Prediction Workshop (DPW)

- The pressure drag convergence of CHAMPS is similar to the workshop results





Applications

Fourth AIAA High Lift Prediction Workshop

Takeoff and Landing aerodynamics simulations remain one of the great challenge of CFD. This workshop gather a community of CFD practitioner from industry and academia to evaluate the state of the art. Participants: NASA, Boeing, JAXA, Embraer, etc.

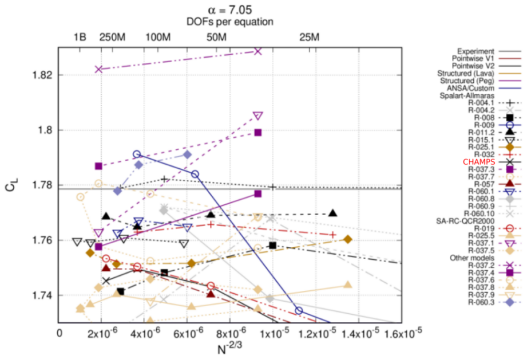
First AIAA Ice Prediction Workshop

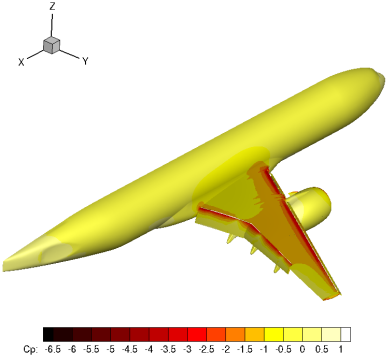
Ice accretion simulations are a major challenge in CFD since it involves multiple physical models working together. 3D simulations are especially difficult due to the complex geometries arising from ice surface topological evolution. This workshop gathered a community of scientists working with icing among the industry, the universities and the research labs. Participants: NASA, ONERA, Boeing, Bombardier, ANSYS, Oxford University, Siemens, FAA, etc.

Application - Fourth AIAA High Lift Prediction Workshop

Fourth AIAA High Lift Prediction Workshop

- Case 1b : Grid refinement study for a constant angle of attack of 7.05°;
- Results are in line with state of the art RANS solver.



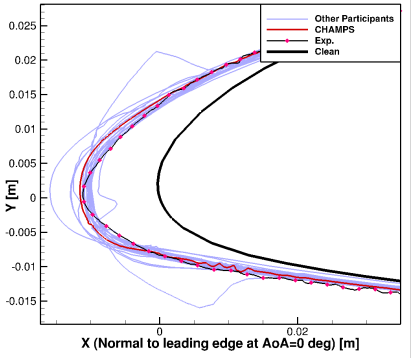


Adapted from Olivier-Gooch, C., Coder, J. 4th CFD High Lift Prediction Workshop, Fixed-Grid RANS TFG, AIAA HLPW4

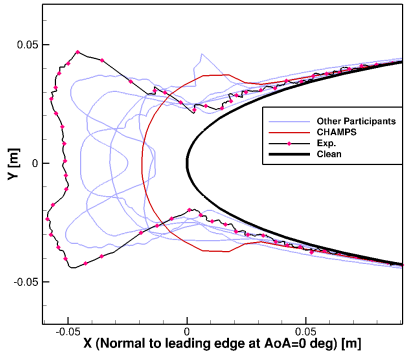
Application - First AIAA Ice Prediction Workshop

First AIAA Ice Prediction Workshop

- Case 241 (left): Rime ice prediction on small NACA23012 airfoil (2D, low temp.);
- Case 363 (right): Glaze ice prediction on NACA0012 swept wing (3D, warmer temp.).



Case 241 (2D rime ice)



Case 363 (3D glaze ice)

Conclusion

Distributed Memory Parallelism

- The development of distributed memory application is efficient
- Complex algorithms are easily portable to large computer clusters

Productivity

- Additional modules are easily added by team members (other than original developers)
- Our experience in writing such software points to 2X-5X faster implementation times for Chapel than C/C++ combined with MPI/OPENMP

Performance

- CHAMPS is performing similarly to other C/C++ applications

Future Work

- Implementation of a fluid-structure interface

Acknowledgements

- This presentation benefited from inputs by S.B. Côté (research Associate) and F. Plante (Post-Doc)
- The authors would like to acknowledge the great support from the Chapel team at Hewlett Packard Enterprise
- This work benefited from the support of: Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada Research Chair Program
- Calculations were performed on Compute Canada/Calcul Quebec clusters
- Large-scale scalability simulations were performed on a cluster provided by Hewlett Packard Enterprise