Development of Parallel CFD Applications with the Chapel Programming Language

Matthieu Parenteau[1], Simon Bourgault-Cote[1], Frederic Plante[1], Engin Kayraklioglu[2] and Éric Laurendeau[1]

1 *Polytechnique Montreal, Quebec, H3T 1J4, Canada*
2 *Hewlett Packard Enterprise, San Jose, CA, 95002, USA*

NSERC CRSNG

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

Table of contents

What is Chapel?*

Chapel

- A modern parallel programming language
- portable & scalable
- open-source & collaborative

Goals

- Support general parallel programming
- Make parallel programming at scale far more productive

* Some of the Chapel slides are based on *Brad Chamberlain, Chapel Comes of Age: A Language for Productivity, Parallelism, and Performance, HPC Knowledge Meeting (HPCKP), 2019* and *Kathy Yelick,CHIUW 2018 keynote: Why Languages Matter More Than Ever*

Why Consider a New Language?

Syntax

- High level, elegant syntax
- **Improve programmer productivity**

Semantics

- Static analysis can help with correctness
- We need a compiler (front-end)

Performance

- **Optimizations are needed to get performance**
- We need a compiler (back-end)

Algorithms

- Language defines what is easy and hard
- **Influences algorithmic thinking**

Chapel

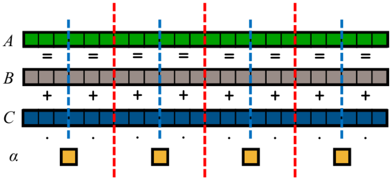
Chapel aims to be as...

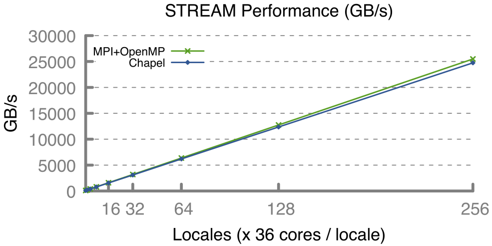
- **...programmable** as Python
- **...fast** as Fortran
- **...scalable** as MPI, SHMEM, or UPC
- **...portable** as C
- **...flexible** as C++

Stream Triad

$$\forall\, i \,\epsilon\, 1..m,\ A_i = B_i + \alpha_i C_i$$

```
use BlockDist;

config const m = 1000;
config const alpha = 3.0;

const Dom = {1..m} dmapped Block ({1..m});

var A, B, C: [Dom] real;

B = 2.0;
C = 1.0;

A = B + alpha * C;
```





- Global namespace is supported (PGAS)
- Chapel uses the concept of *locale*

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

CHAMPS

Objective

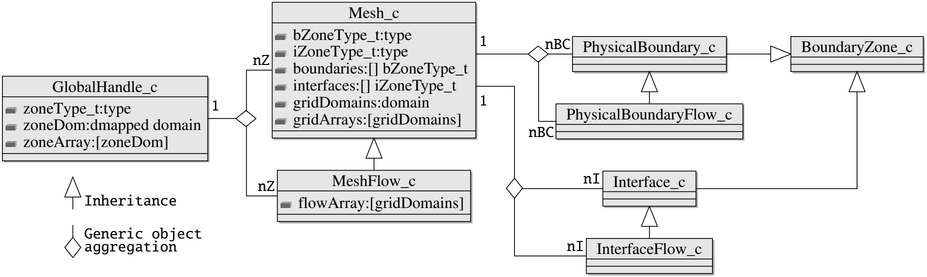
Build a complete 3D unstructured CFD simulation software from scratch with Chapel for large-scale simulation on distributed memory for multi-physics simulations.

CHapel Multi-Physics Simulation (CHAMPS)

- 3D Unstructured Reynolds Average Navier-Stokes flow solver (support 2D and 3D grids)
- Second order finite volume
- Convective flux schemes: Roe and AUSM
- SA, $k - \omega$ SST-V and Langtry-Menter transitional turbulence models
- Explicit solver (Runge Kutta) and implicit solvers (SGS, GMRES)
- Jacobian-Free Newton-Krylov
- Interface with external libraries: MKL, CGNS, METIS and PETSC
- Icing module (not shown)

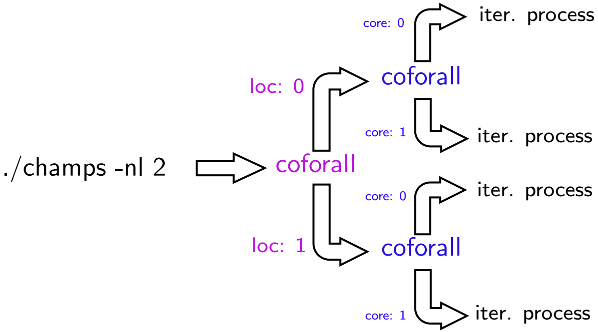
CHAMPS

- A Multi-physics problem requires different computational grids
- Type aliases are used to define these various computational domains



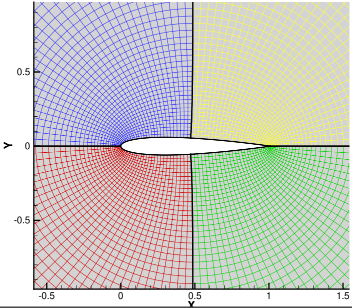
Parallelism over distributed memory

- Single execution
- A task is created for every available *locale*
- The task on a *locale* creates a task for every core on that *locale*



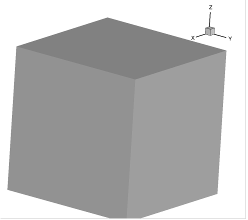
Communication

```
proc performInterfaceExchanges(zone, exchangeType)
{
    // Fill buffers
    local do zone.prepareExchange(exchangeType);
    allLocalesBarrier.barrier();

    // Read buffers
    zone.exchangeInterfaces(exchangeType);
    allLocalesBarrier.barrier();
}
```



Scalability

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



Strong Scaling

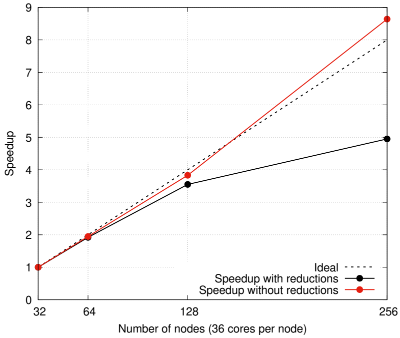
The cube is discretized with 800 elements in every direction (i,j,k) for a total of 512M elements

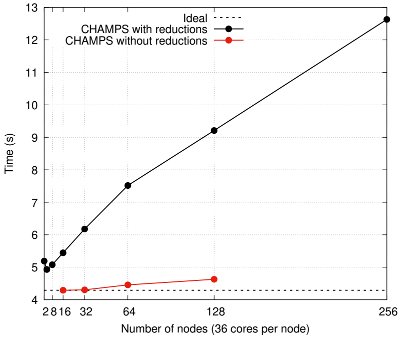
Weak Scaling

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



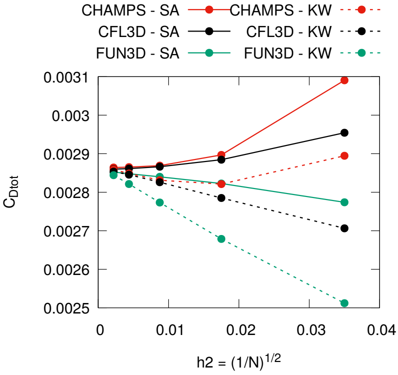


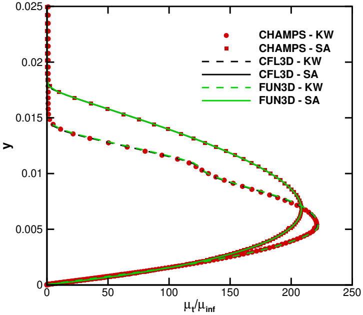
(Left) - Strong Scaling (Right) - Weak Scaling

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

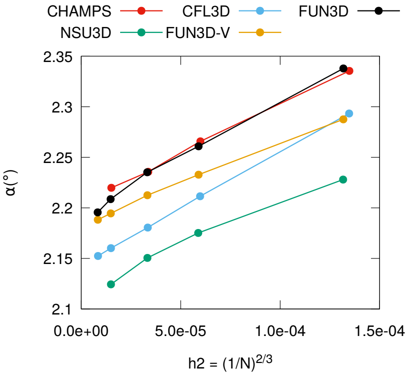


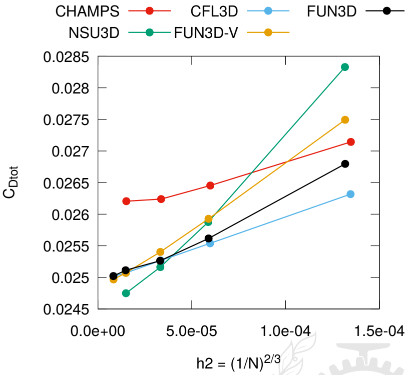


Numerical Verification

Fifth Drag Prediction Workshop (DPW)

- The angle of attack to yield a lift coefficient of 0.5 is in line with the workshop results

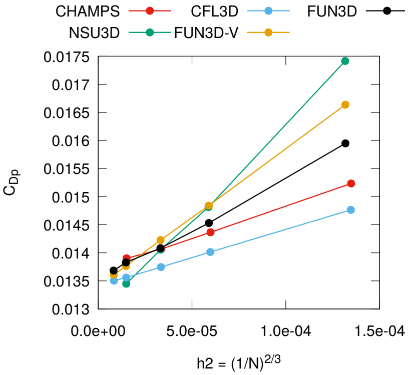


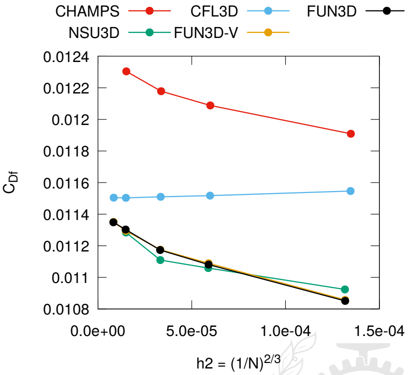


Numerical Verification

Fifth Drag Prediction Workshop (DPW)

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





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 2-5X faster implementation times for Chapel than C/C++

Performance

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

Future Work

- Implementation of a fluid-structure interface

Acknowledgements

- 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), MITACS 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