

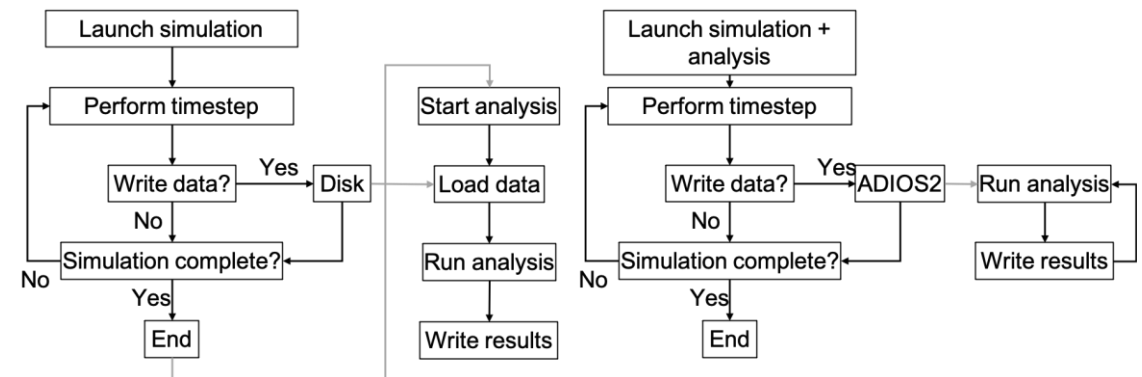
ARCHER2 ECSE03-2 DEVELOPING IN-SITU ANALYSIS CAPABILITIES FOR PRE-EXASCALE SIMULATIONS WITH XCOMPACT3D

Paul Bartholomew¹, Charles Moulinec², Michèle Weiland¹, Sylvain Laizet³ (PI)

1) EPCC, 2) STFC, 3) Imperial College London

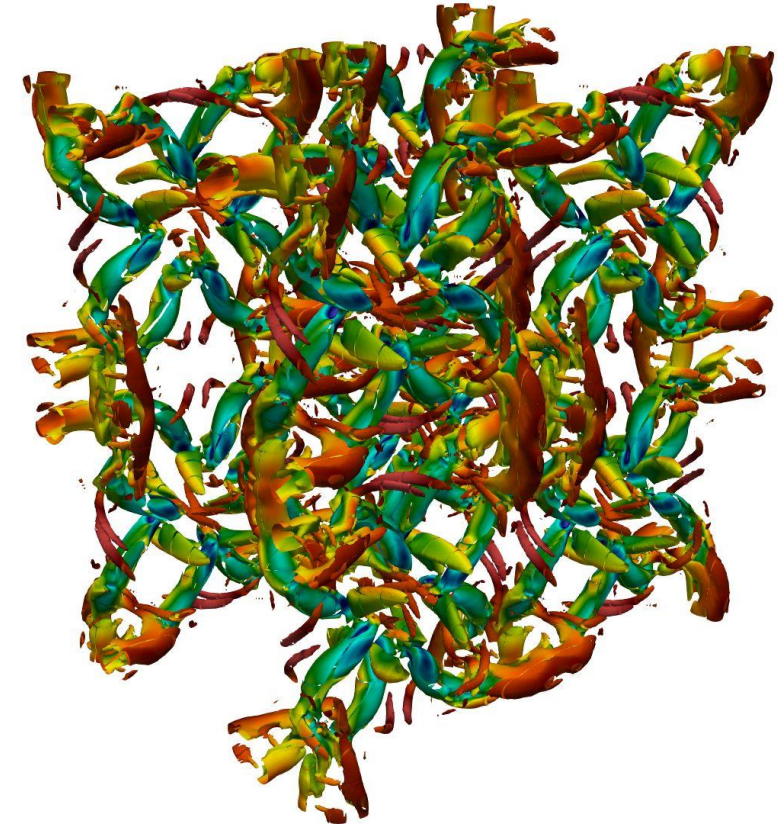
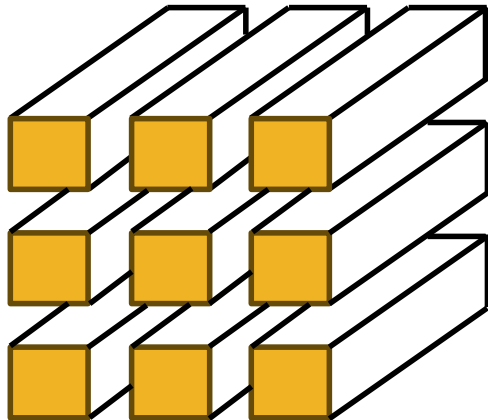
Project overview

- Add ADIOS2 as an (optional) I/O backend to 2DECOMP&FFT
- Parallelise Py4Incompact3d post-processing suite
 - Wrap DECOMP&FFT for parallelisation and (some) I/O
- Use python interface to ADIOS2 in Py4Incompact3d
- Couple Xcompact3d and Py4Incompact3d via ADIOS2 for *in-situ* analysis
 - Perform data reduction online
 - Reduce disk bandwidth requirements
 - Eliminate I/O bottleneck
 - Enable/simplify high-frequency data analysis for users



Xcompact3d

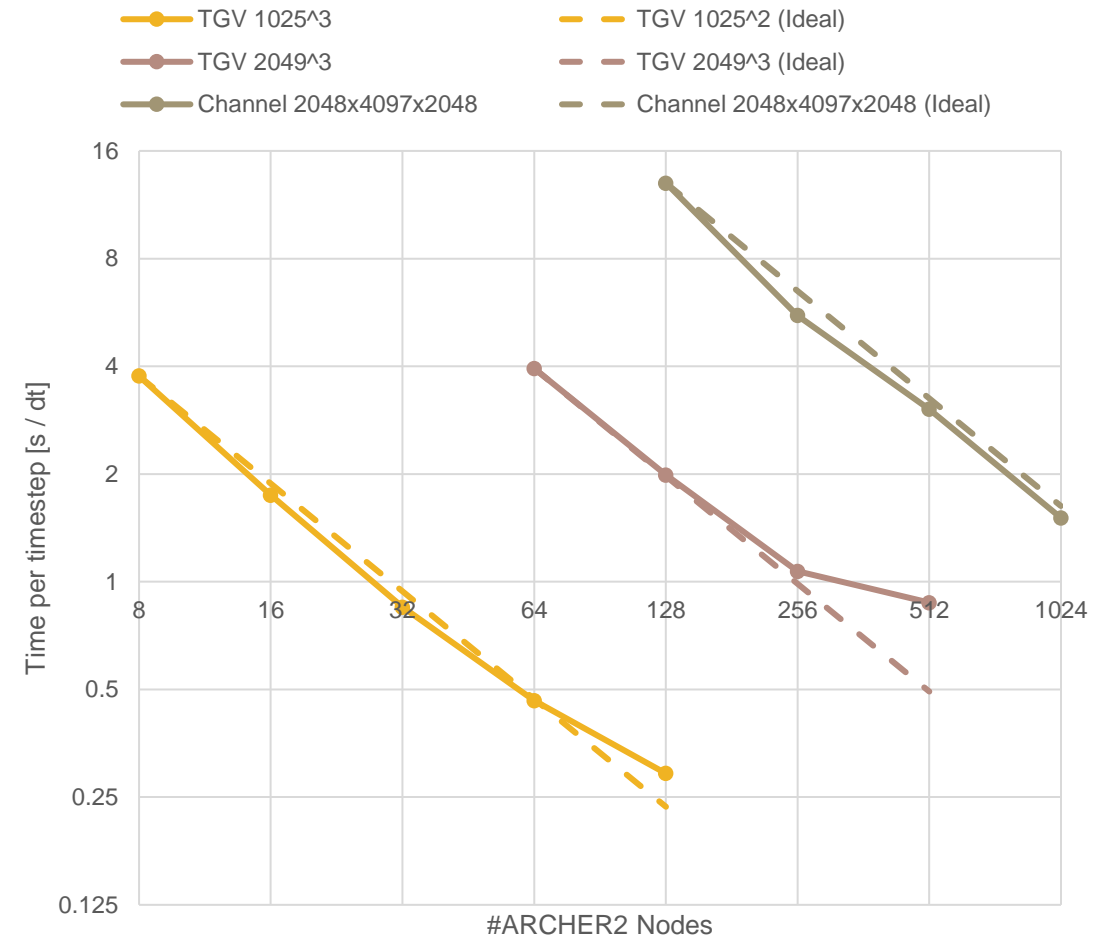
- CFD code based on high-order compact finite difference schemes + spectral Poisson solver
 - Global developer and user base
- Parallelised using 2-D “pencil” domain decomposition
- Parallel operations inc. I/O and FFT interface via 2DECOMP&FFT library



Visualisation of Taylor-Green Vortex simulated with Xcompact3d.

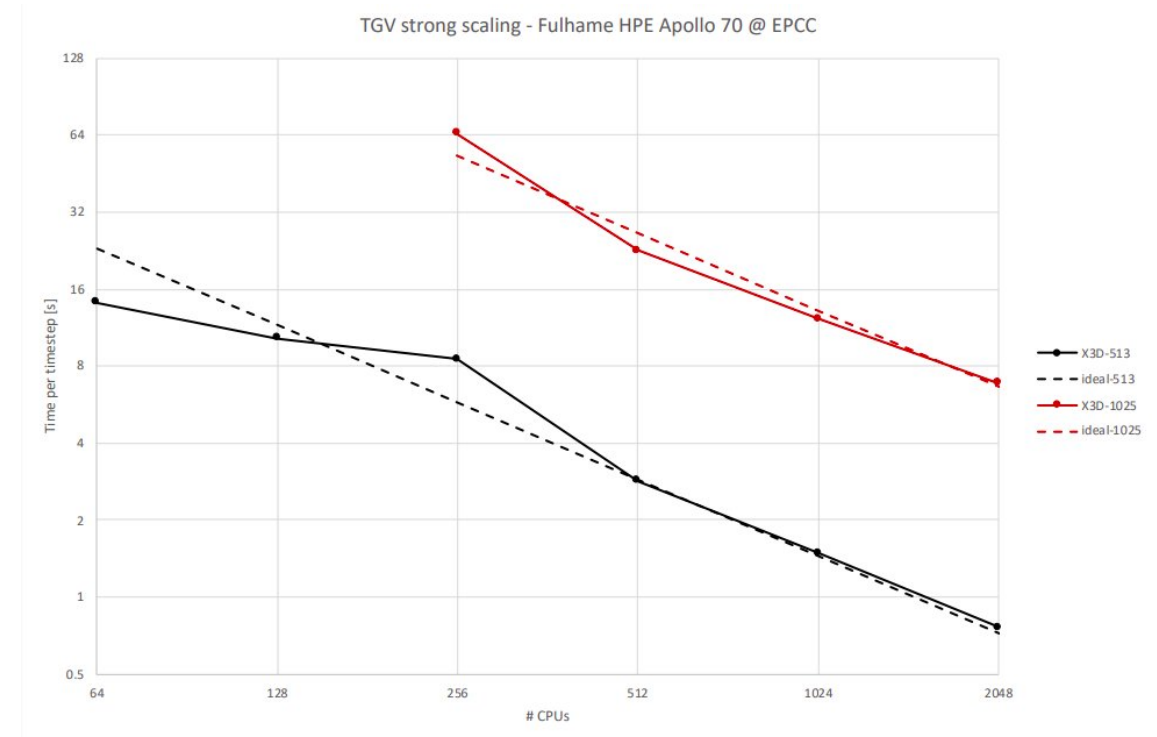
Motivation: Compute vs I/O performance

- Solvers receive lots of attention
 - Xcompact3d demonstrated scaling up to $\mathcal{O}(10^6)$ MPI ranks



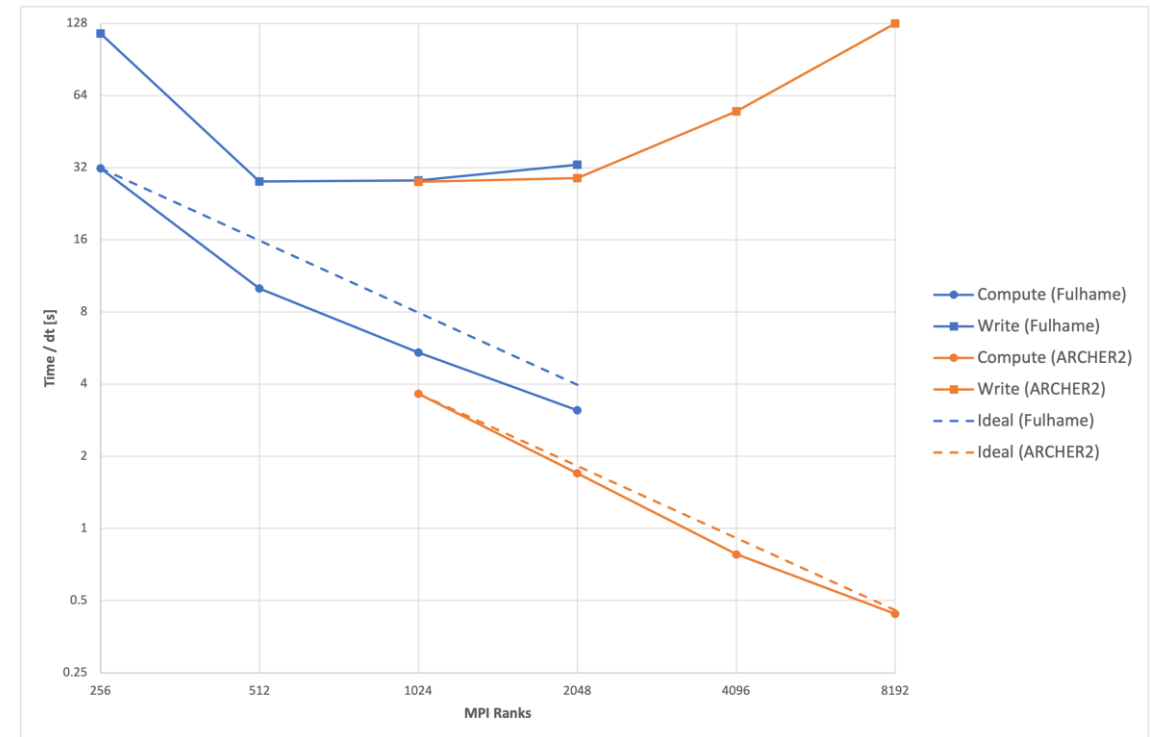
Motivation: Compute vs I/O performance

- Solvers receive lots of attention
 - Xcompact3d demonstrated scaling up to $\mathcal{O}(10^6)$ MPI ranks
 - Portable across CPU architectures



Motivation: Compute vs I/O performance

- Solvers receive lots of attention
 - Xcompact3d demonstrated scaling up to $\mathcal{O}(10^6)$ MPI ranks
 - Portable across CPU architectures
- General scientific workflow
 1. Run simulation
 - Output solution
 2. Analyse results
 3. Answer question!
- Required resolution vs cost
 - I/O performance often overlooked
 - Volume of storage
 - Data management



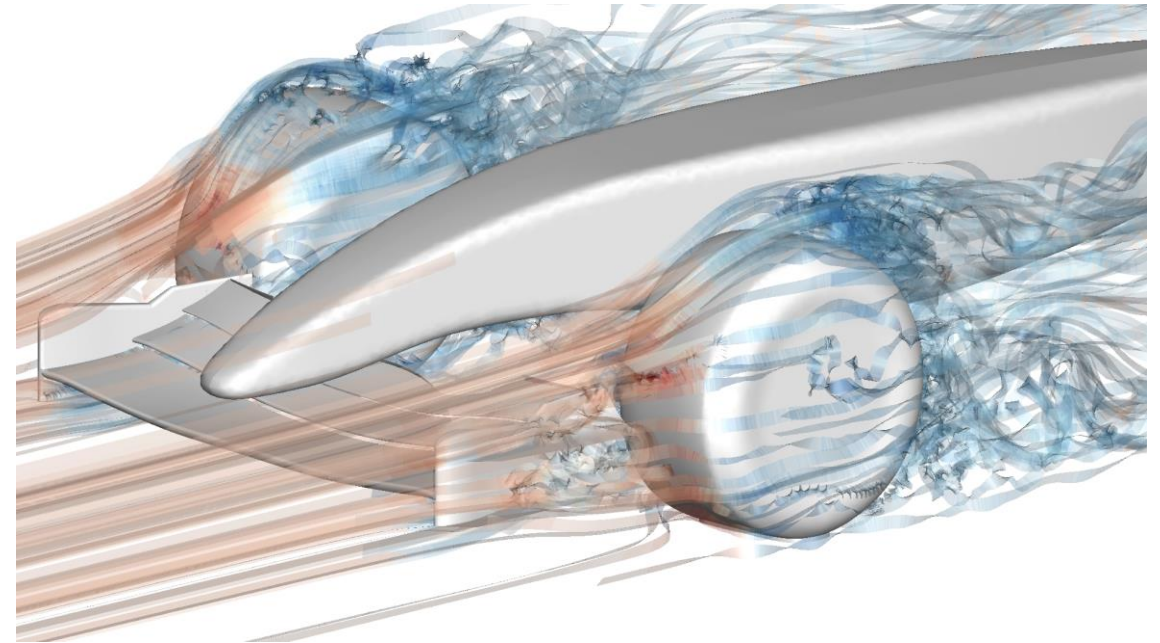
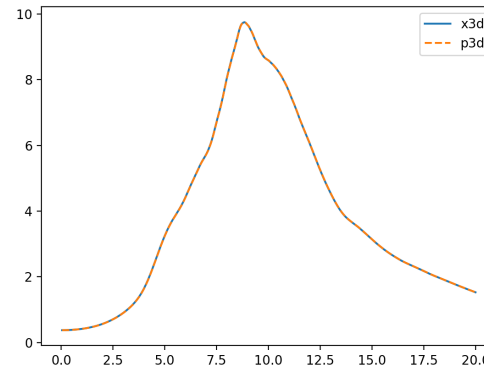
Xcompact3d compute vs I/O breakdown. **N.B.** pre-production ARCHER2 system!

In-situ analysis

- Simulations aim to answer a question
- Make some measurement of solution

$$drag = \int_{\partial\Omega} f(u) dS$$

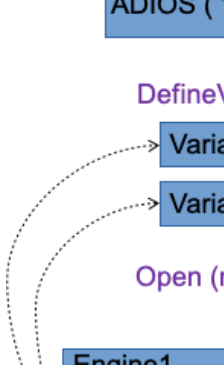
- Often required data \ll full 3-D field
 - Full 3-D field is still useful!
- *In-situ* analysis performs online data reduction
 - Often necessary for high-resolution data

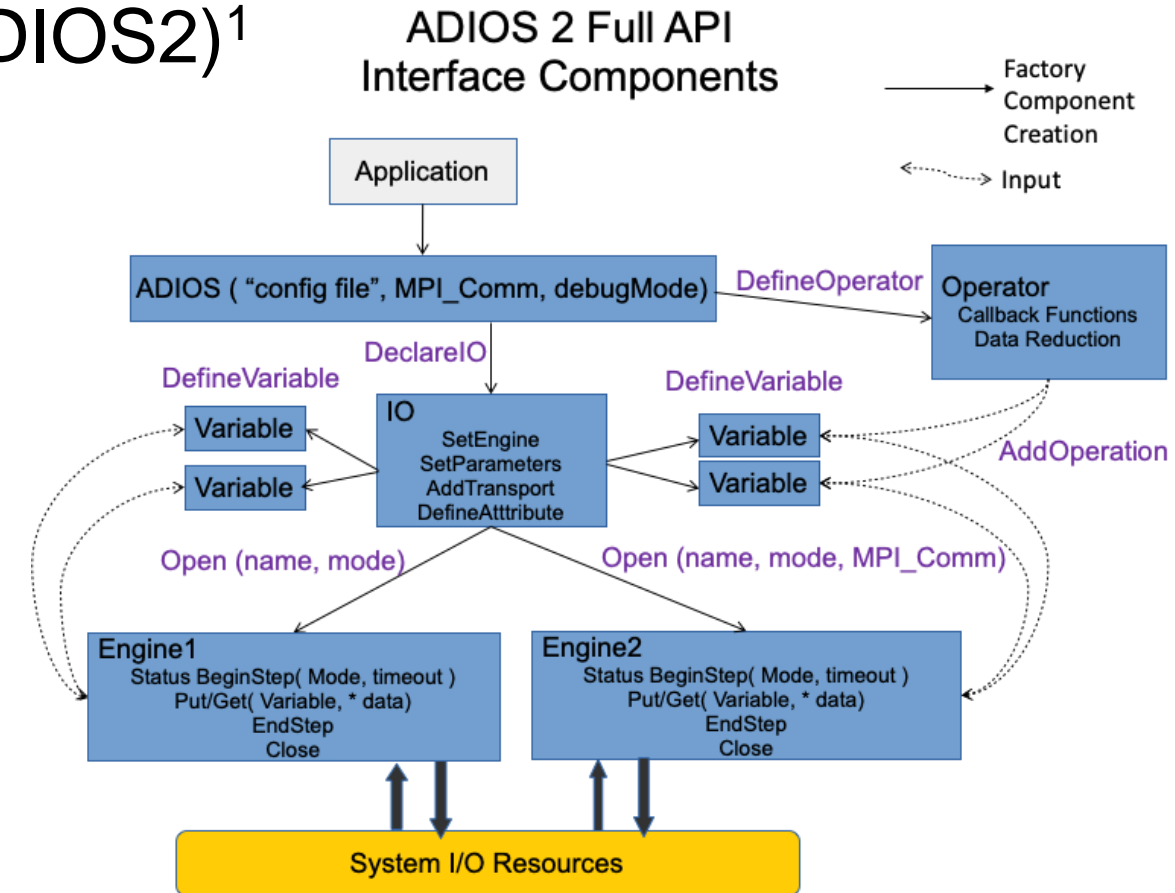


Flow around F1 car. J-E. Lombard (2015)

<https://www.nektar.info/formula-1/>

ADIOS2

- ADaptable Input Output System (ADIOS2)¹
 - C++ with C/Fortran/Python bindings
 - Runtime configurable
 - Select I/O format
 - Select I/O *destination*
 - File, network
 - Extensible via plugins
- 
- ```
graph TD; ADIOS[ADIOS (User)] -- DefineVariables --> Variables[Variable Objects]; Variables -- Open (r/w) --> Engine1[Engine1];
```





# Integrating ADIOS2 into Xcompact3d

- Goal to support existing (MPI-IO) workflow + new ADIOS2 backend
- Selected at compile time via preprocessor macros

```
#ifndef ADIOS2
```

```
 ! Existing MPI-IO code...
```

```
#else
```

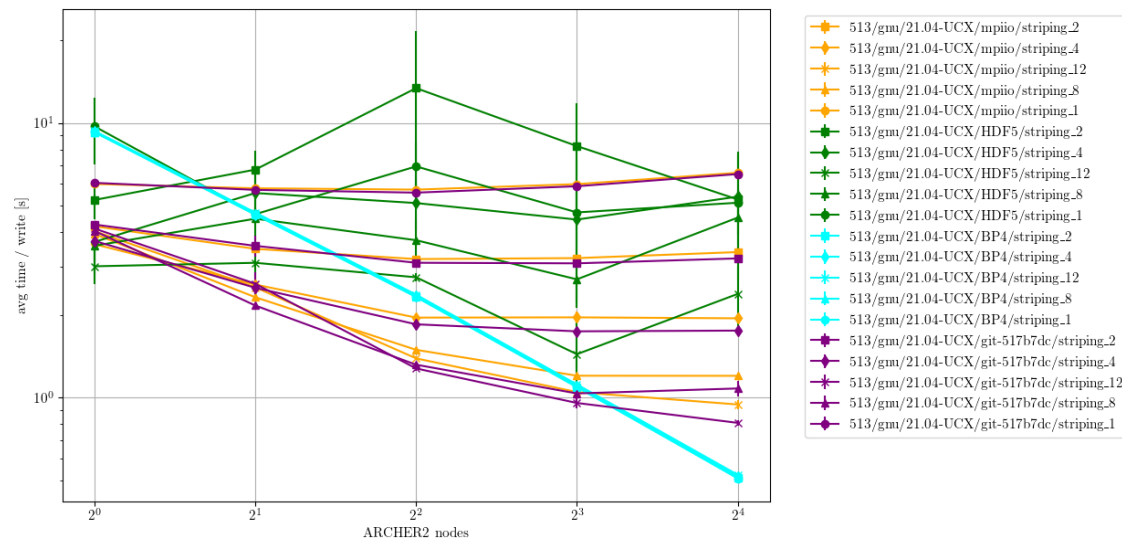
```
 ! New ADIOS2 call
```

```
#endif
```

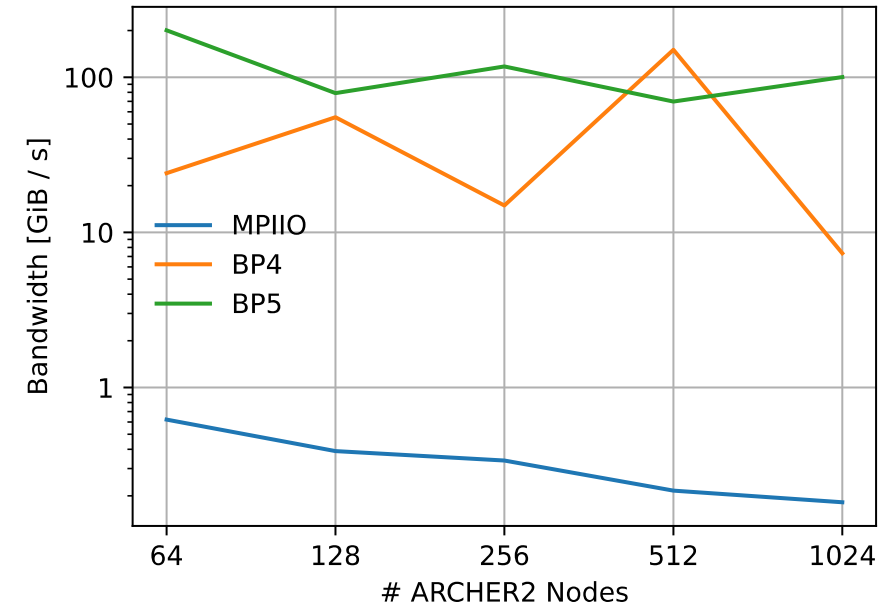
- With hindsight, this was a mistake
- Changes primarily in backend library (2DECOMP&FFT)
  - “Pre-registration” of data (noop in MPI-IO)
  - (Multiple) files per step vs multiple steps per file
  - I/O buffering in ADIOS2

```
<?xml version="1.0"?>
<adios-config>
 <io name="solution-io">
 <engine type="HDF5">
 <parameter key="H5CollectiveMPIIO" value="yes"/>
 </engine>
 </io>
 <io name="restart-io">
 <engine type="HDF5">
 <parameter key="H5CollectiveMPIIO" value="yes"/>
 </engine>
 </io>
 <io name="statistics-io">
 <engine type="HDF5">
 <parameter key="H5CollectiveMPIIO" value="yes"/>
 </engine>
 </io>
 <io name="restart-forces-io">
 <engine type="HDF5">
 <parameter key="H5CollectiveMPIIO" value="yes"/>
 </engine>
 </io>
</adios-config>
```

# Comparing ADIOS2 and MPI-IO



Runtime on ARCHER2 across backends (TGV 512<sup>3</sup>)



Achieved I/O bandwidth on ARCHER2 (TGV 1025<sup>3</sup>)

## Py4Incompact3d

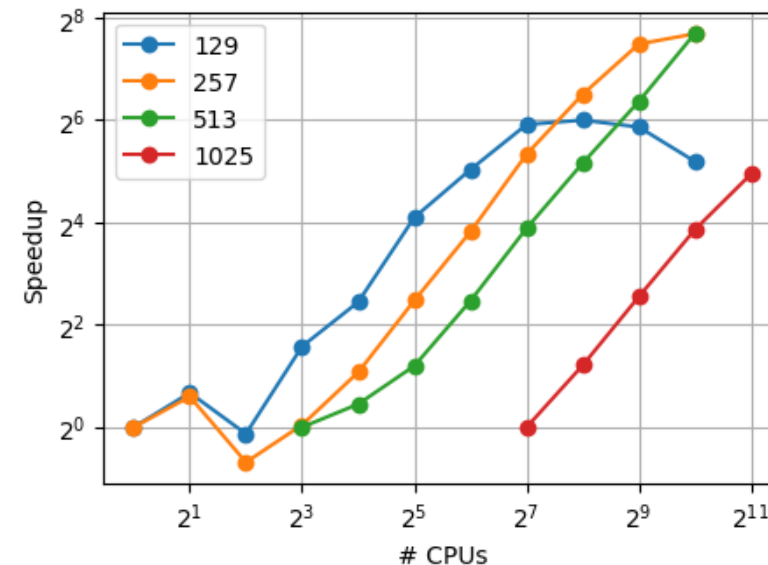
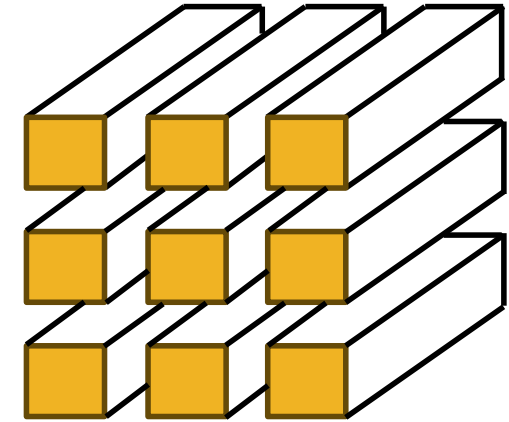
- Collaborative effort to combine postprocessing scripts
- Based on numpy
- Simplifies reading data generated by Xcompact3d
  - Parsing diagnostics
  - Loading series of 3-D datasets
- Implements the same compact schemes as Xcompact3d

$$\alpha f'_{i-1,j,k} + \alpha f'_{i,j,k} + \alpha f'_{i+1,j,k} = R(f_{s(i),j,k})$$

- Numpy does the heavy lifting

## Parallelising Py4Incompact3d

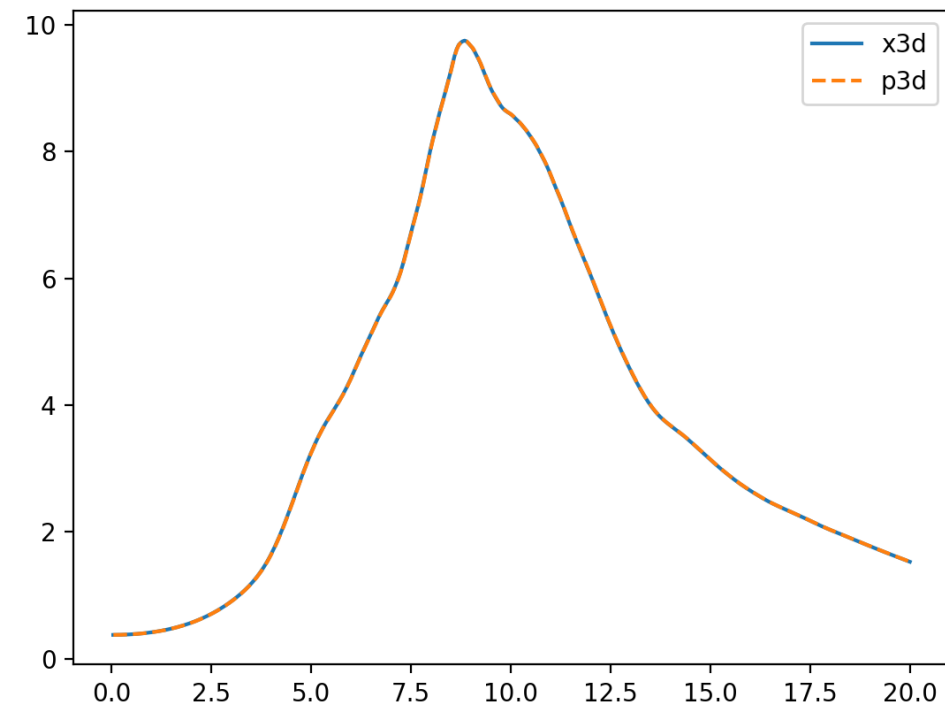
- Initially developed for workstation use (serial)
- Same numerical schemes as Xcompact3D
  - Same parallelisation strategy should work
- Exposed Python bindings to 2DECOMP&FFT via f2py
- Simplified parallel strategy
  - Ease of use
  - Working direction always “X”
    - Adds additional overheads



Computing  $\nabla\phi$  using  
Py4Incompact3D on ARCHER2

## *In-situ* analysis with ADIOS2 and Py4Incompact3d

- Use ADIOS2 “high-level” Python interface
- Using ADIOS2 “simply” switch I/O layer to SST
  - Runtime configuration
- Needed to work around the Python/ADIOS2 interface
  - Not ideal
  - Expected to be stable...
- Results match with Xcompact3D
- Producer (Xcompact3D) outpacing consumer (Py4Incompact3D) leads to memory pile up
  - Eventually OOM!



# Optimising Py4Incompact3d

- Naïve TDMA implementation used Python outer loop
  - Inner loops numpy-vectorised
  - Too slow!
- Numpy cannot express loop-carried dependency
- Need to rethink algorithm
  - Express in pure numpy

- Variable coefficient inhomogeneous recurrence relation at core

$$x_{i+1} = a_i x_i + b_i$$

- Has a closed form solution<sup>1</sup>

$$x_i = \left( \prod_{j=0}^{i-1} a_j \right) \left( X_0 + \sum_{j=0}^{i-1} \frac{b_j}{\prod_{k=0}^j a_k} \right)$$

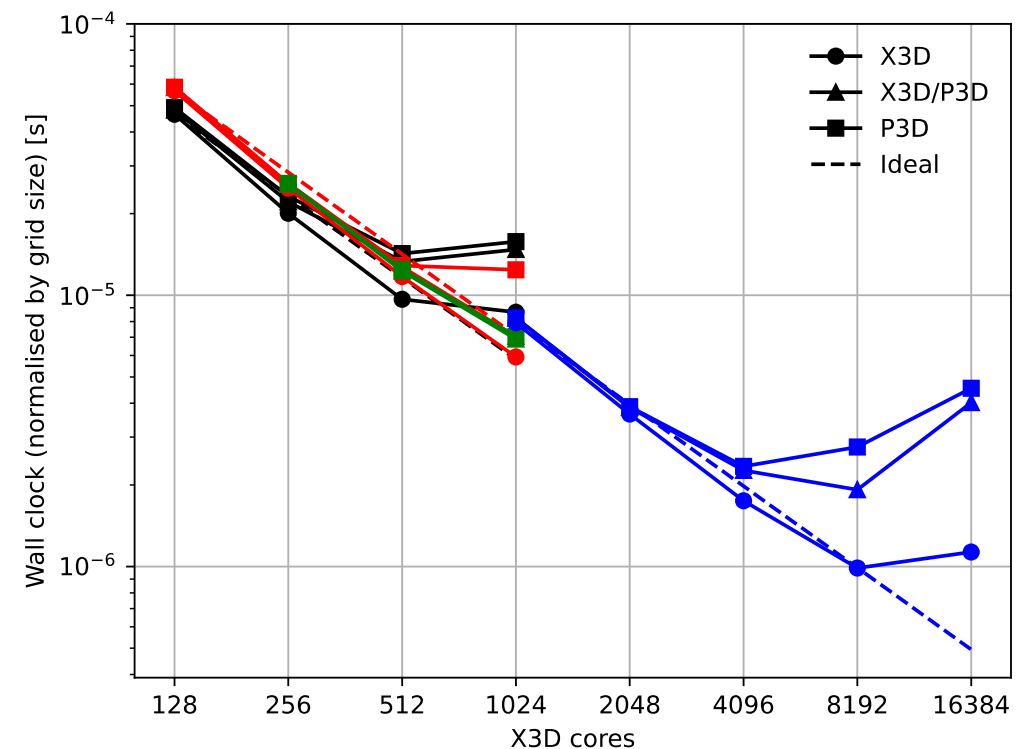
- Can be written in numpy/linalg
- Keeps up with Xcompact3D in testing

[1] Solving first-order non-homogeneous recurrence relations with variable coefficients

[https://en.wikipedia.org/wiki/Recurrence\\_relation](https://en.wikipedia.org/wiki/Recurrence_relation)

## Demonstrating Xcompact3D+Py4Incompact3D vs Xcompact3D standalone

- Taylor Green Vortex case
  - X3D on 1-128xARCHER2 nodes
  - Using optimised Py4Incompact3D
- Range of problem sizes
  - $129^3$  mesh, P3D on 1xARCHER2 nodes (black)
  - $257^3$  mesh, P3D on 1xARCHER2 nodes (red)
  - $257^3$  mesh, P3D on 2xARCHER2 nodes (green)
  - $513^3$  mesh, P3D on 8xARCHER2 nodes (blue)
- Observe minimal slowdown until > 4:1 X3D:P3D cores ratio



## Conclusion & Future work

- New (optional) ADIOS2 I/O backend added to Xcompact3D (2DECOMP&FFT)
- Immediate user benefits in I/O performance
- Py4Incompact3D parallelised for HPC use
- *In-situ* analysis demonstrated with concurrent Xcompact3D+Py4Incompact3D on ARCHER2
- New version of 2DECOMP&FFT released (v2.0 – published in JOSS)
- Improved object-oriented I/O interface under development by 2DECOMP&FFT team
  - Addresses shortcomings of the macro-based approach
- Investigate ADIOS2 file compression
- *In-situ* visualisation using ParaView/Catalyst



## References

- P. Bartholomew *et al.* “*Xcompact3D: An open-source framework for solving turbulence problems on a Cartesian mesh*”, SoftwareX (2020)
- S. Rolfo *et al.* “*The 2DECOMP&FFT library: An update with new CPU/GPU capabilities*”, JOSS (2023)
- P. Bartholomew *et al.* “*Developing in-situ analysis capabilities for pre-Exascale simulations with Xcompact3D*”, tech. rep. ARCHER2 eCSE03-2 (2023), <https://www.archer2.ac.uk/ecse/reports/eCSE03-02/>
- Xcompact3D: <https://github.com/xcompact3d/Incompact3d>
- Py4Incompact3D: <https://github.com/xcompact3d/Py4Incompact3D>
- 2DECOMP&FFT: <https://github.com/2decomp-fft/2decomp-fft>