A review of OpenFOAM collated I/O performance on ARCHER

OpenFOAM

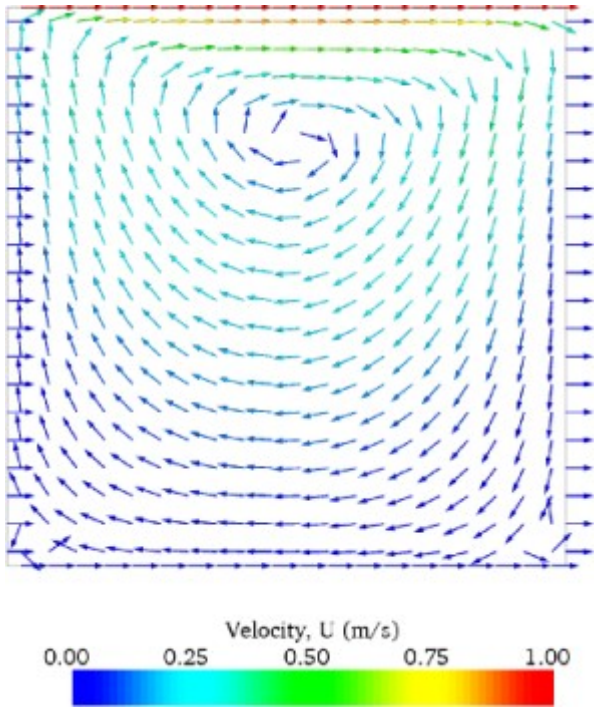
- Open-source Field Operation and Manipulation
- Free computational fluid dynamics software
- Written in C++
- Efficiently parallelised
- Widely-used application on ARCHER

The file problem

OpenFOAM creates one file per physical property per processor per timestep!!

resources	**Resource Pool**				**Remaining Budget**
	XC				5,212.7 kAUs
	Volume	**Usage**	**Quota**	**Files**	**File Quota**
	home (home4)	5 GiB	1,000 GiB		
	work (fs2)	284 GiB	500 GiB	112,008 Files	500,000 Files

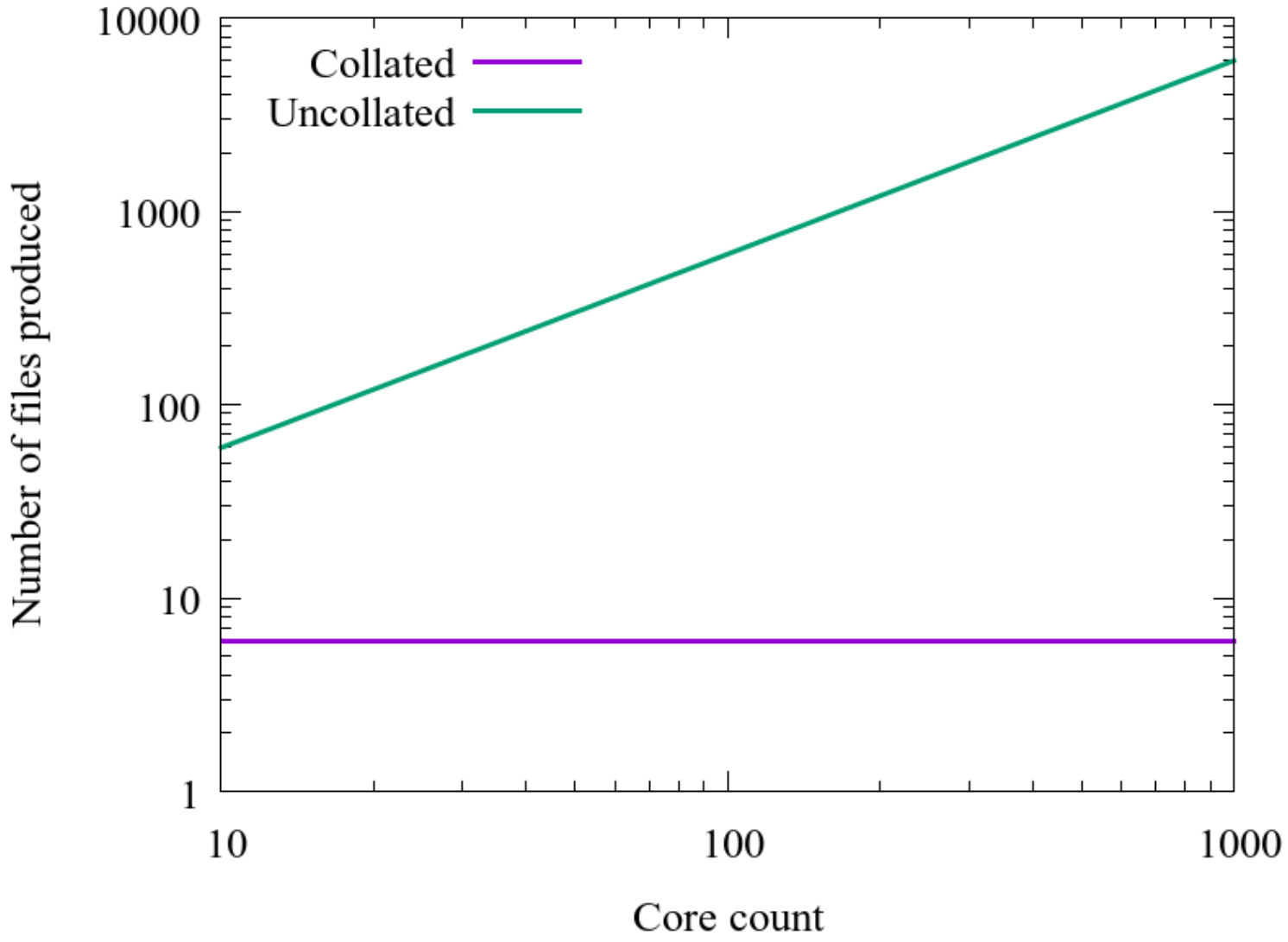
The system



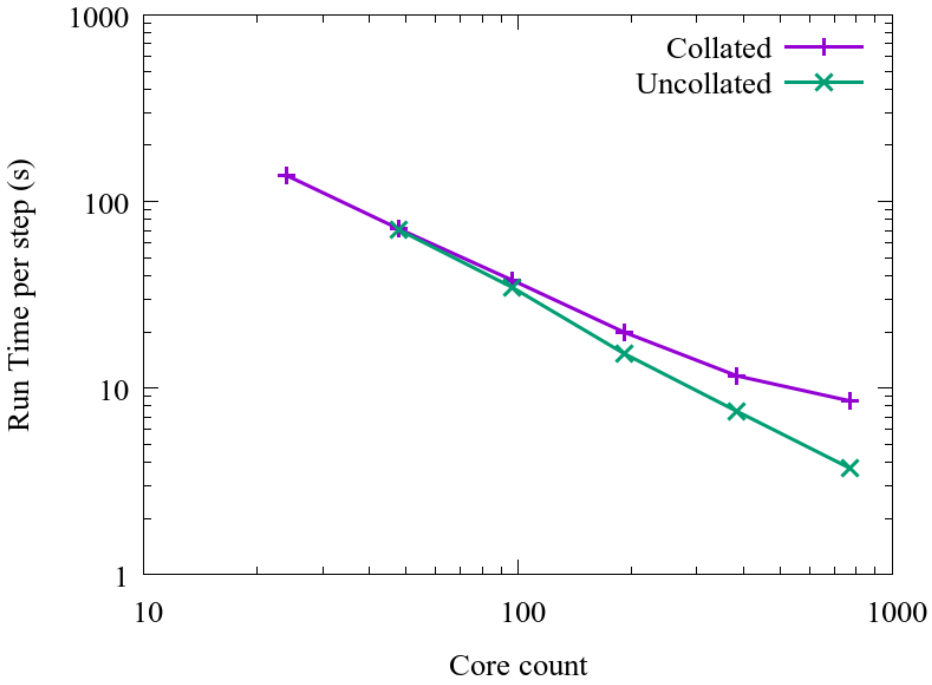
- Using ‘Lid-driven cavity flow’ example
- Dimensions changed from $0.10 \times 0.10 \times 0.01 m^3$ to $2.0 \times 1.5 \times 1.0 m^3$
- 200 grid points per meter
- Tracks 6 physical properties
- 10 runs of 10 outputs each
- At write-precision of 6, produces 2 GB of data

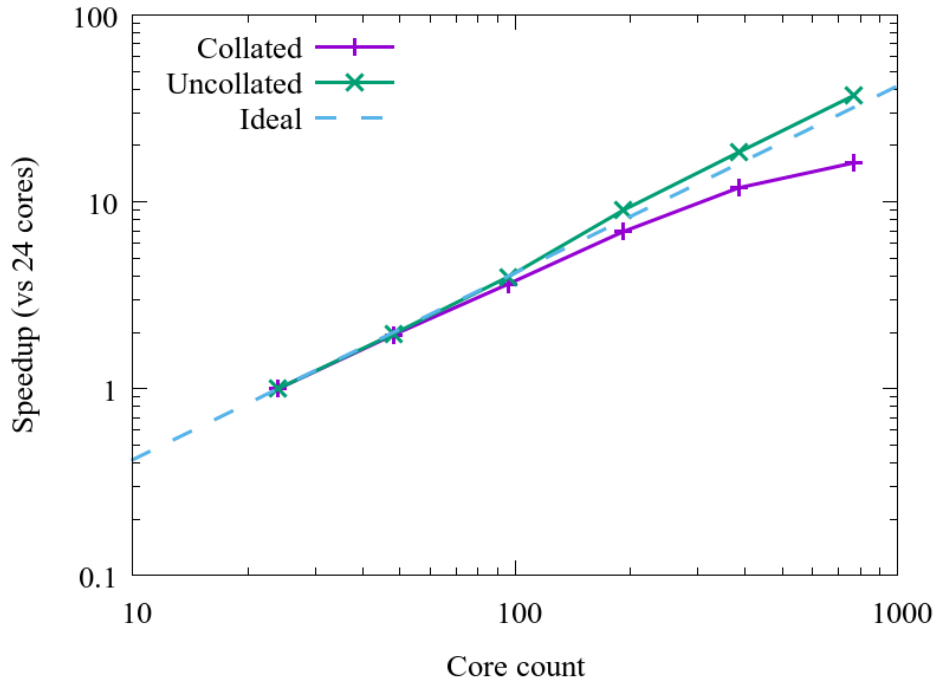
https://www.openfoam.com/documentation/tutorial-guide/tutorialse2.php#x6-60002.1

File number


10000
1000
100
10
1
Collated
Uncollated
Number of files produced
10
100
1000
Core count

Effects on performance



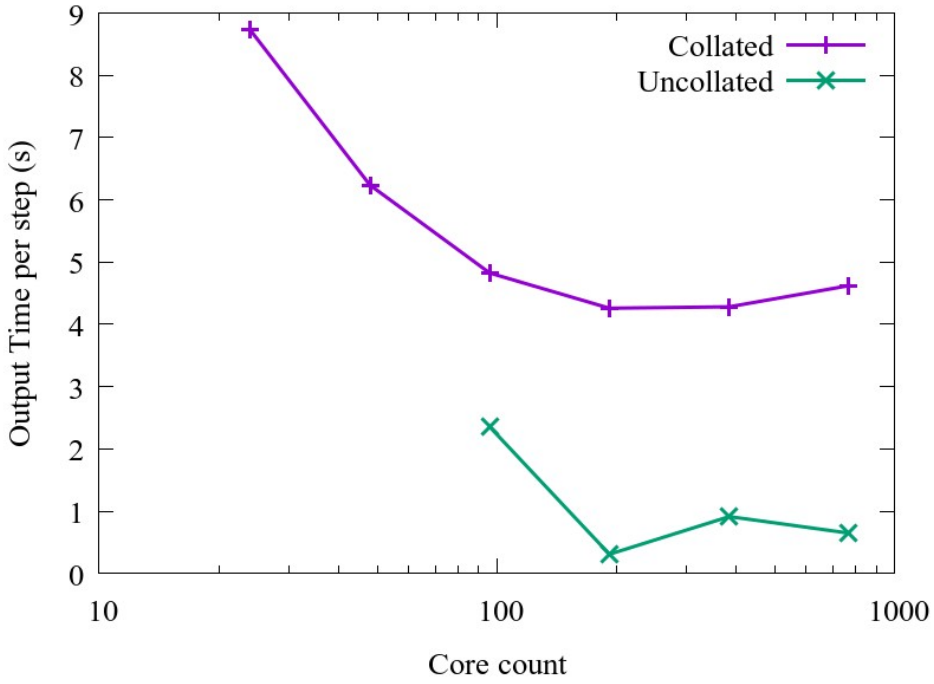


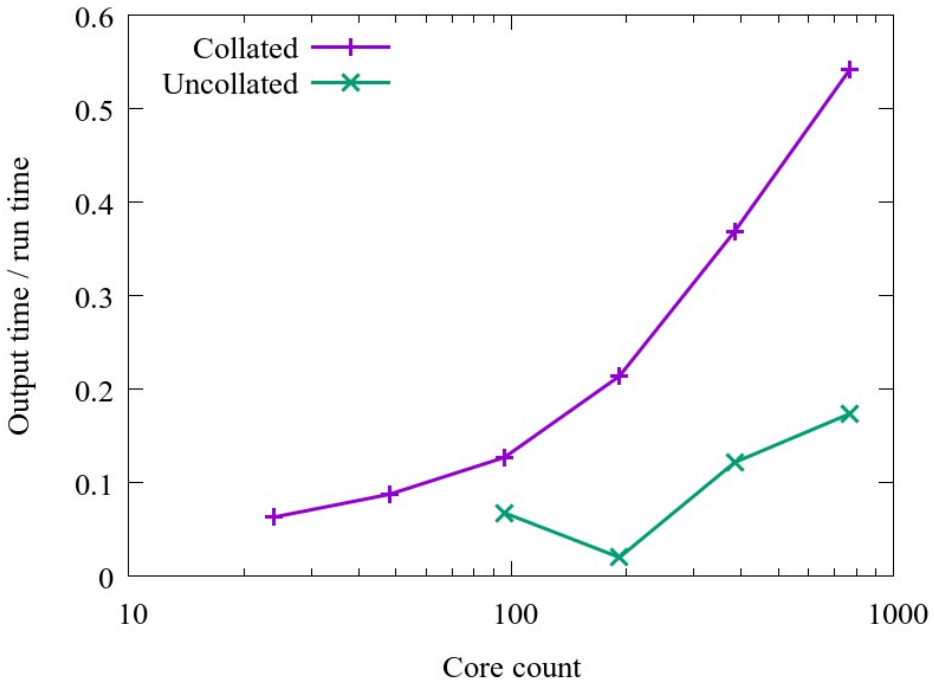
Profiling OpenFOAM

```
Each sample counts as 0.01 seconds.
 no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
 0.00      0.00     0.00 10500000     0.00     0.00  Foam::Ostream& Foam::operator<< <Foam::Vector<double>, double
 0.00      0.00     0.00  1030301     0.00     0.00  Foam::Istream& Foam::operator>><Foam::Vector<double>, double,
 0.00      0.00     0.00     9407     0.00     0.00  frame_dummy
 0.00      0.00     0.00     6673     0.00     0.00  Foam::word::stripInvalid()
 0.00      0.00     0.00     6494     0.00     0.00  Foam::OPstream::~OPstream()
 0.00      0.00     0.00     3904     0.00     0.00  Foam::tmp<Foam::Field<double> >::cref() const
 0.00      0.00     0.00     3273     0.00     0.00    gnu cxx::  enable if<std::  is char<char>::  value, bool>::
 0.00      0.00     0.00     2856     0.00     0.00  Foam::tmp<Foam::fvsPatchField<double> >::ptr() const
 0.00      0.00     0.00     2387     0.00     0.00  Foam::List<double>::List(int)
 0.00      0.00     0.00     2112     0.00     0.00  Foam::fvsPatchField<double>::New(Foam::word const&, Foam::wor
 0.00      0.00     0.00     1762     0.00     0.00  Foam::tmp<Foam::Field<double> >::ref() const
 0.00      0.00     0.00     1432     0.00     0.00  Foam::tmp<Foam::Field<double> >::tmp(Foam::Field<double>*)
 0.00      0.00     0.00     1120     0.00     0.00  void Foam::fvMatrix<Foam::Vector<double> >::addToInternalFiel
 0.00      0.00     0.00      840     0.00     0.00  Foam::word::word(char const*, bool)
 0.00      0.00     0.00      800     0.00     0.00  std::__cxx11::basic_string<char, std::char_traits<char>, std:
 0.00      0.00     0.00      754     0.00     0.00  Foam::word::word(std::__cxx11::basic_string<char, std::char_t
 0.00      0.00     0.00      720     0.00     0.00  Foam::fvPatchField<double>::New(Foam::word const&, Foam::word
```

Getting output time





Conclusions

- At node counts $N < 8$ (core count < 192), both methods are similar
- At node counts $N > 16$ (core count < 384), file-per-process seems better
  - But there is hidden cost of re-combining data
- Still some work to be done here
  - Larger node count
  - Effect of file size
  - *Etc*.

Thanks for your time. Any questions?

Thanks also to:

- Juan Rodriguez-Herrera and Clair Barrass for running the show behind the scenes
- Adrian Jackson for putting up with my strange OpenFOAM questions