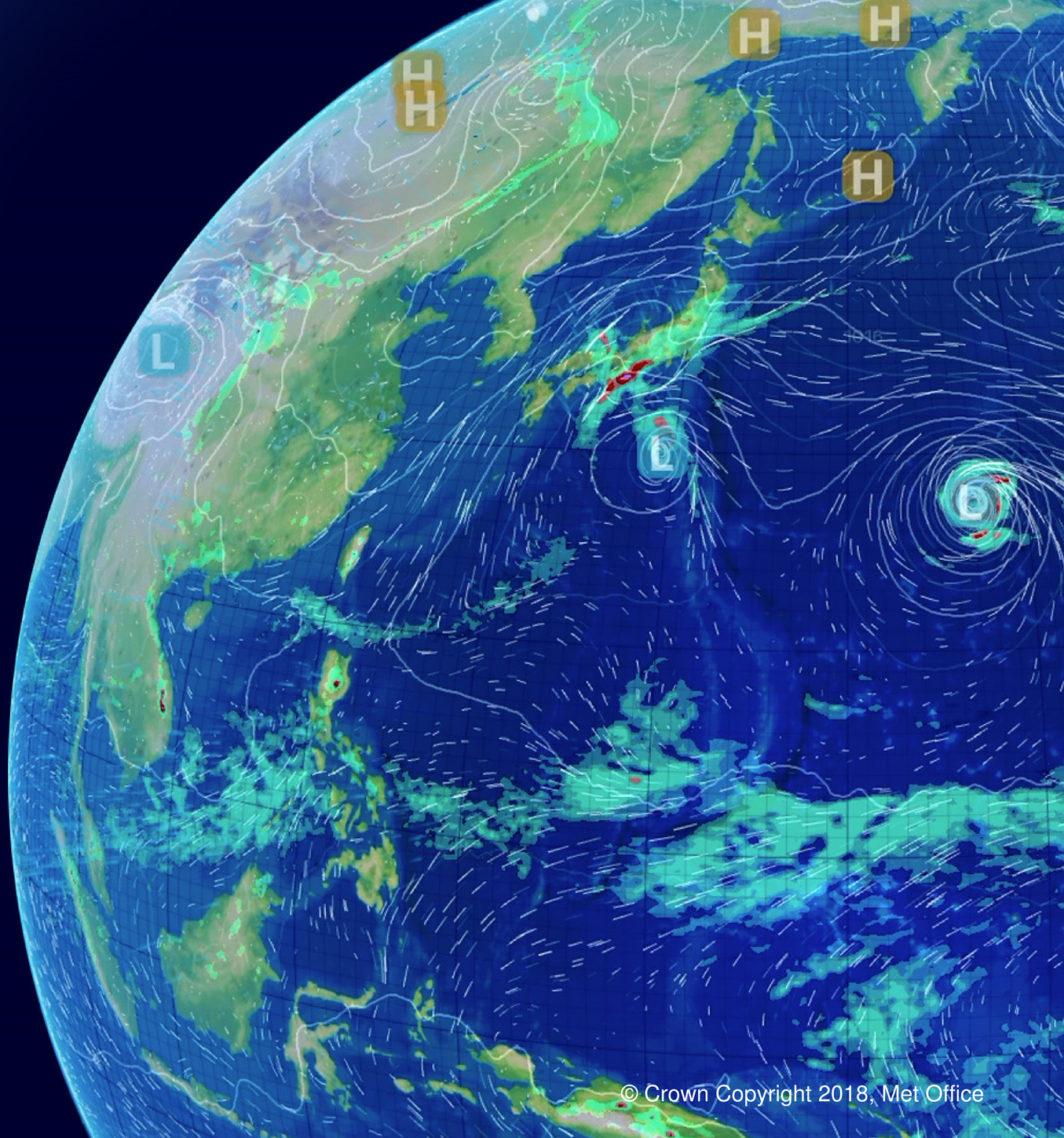
Dealing with I/O performance issues within the Unified Model

SIG-IO-UK - Wed June 6th, 2018

Adam Voysey,

HPC Optimisation Team



Met Office

A Quick introduction to I/O in the Unified Model

The UM

The Unified Model (UM) is a numerical model of the atmosphere used for both weather and climate applications. It is in continuous development by the Met Office and its partners, adding state of the art understanding of atmospheric processes to new releases.

For more information see:
https://www.metoffice.gov.uk/research/modelling-systems/unified-model

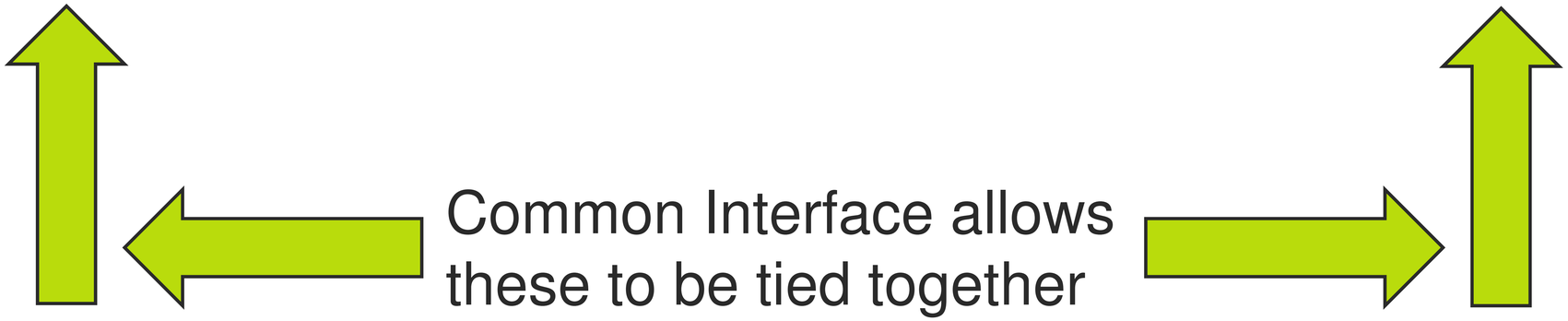
Two Key I/O Components

I/O Server (Remote I/O)

- Written in Fortran
- OpenMP + MPI Parallelism
- Multiple “Roles”

Portio (Local I/O)

- Written in C
- Serial Code*
- Multiple “Layers”



Two Key I/O Components

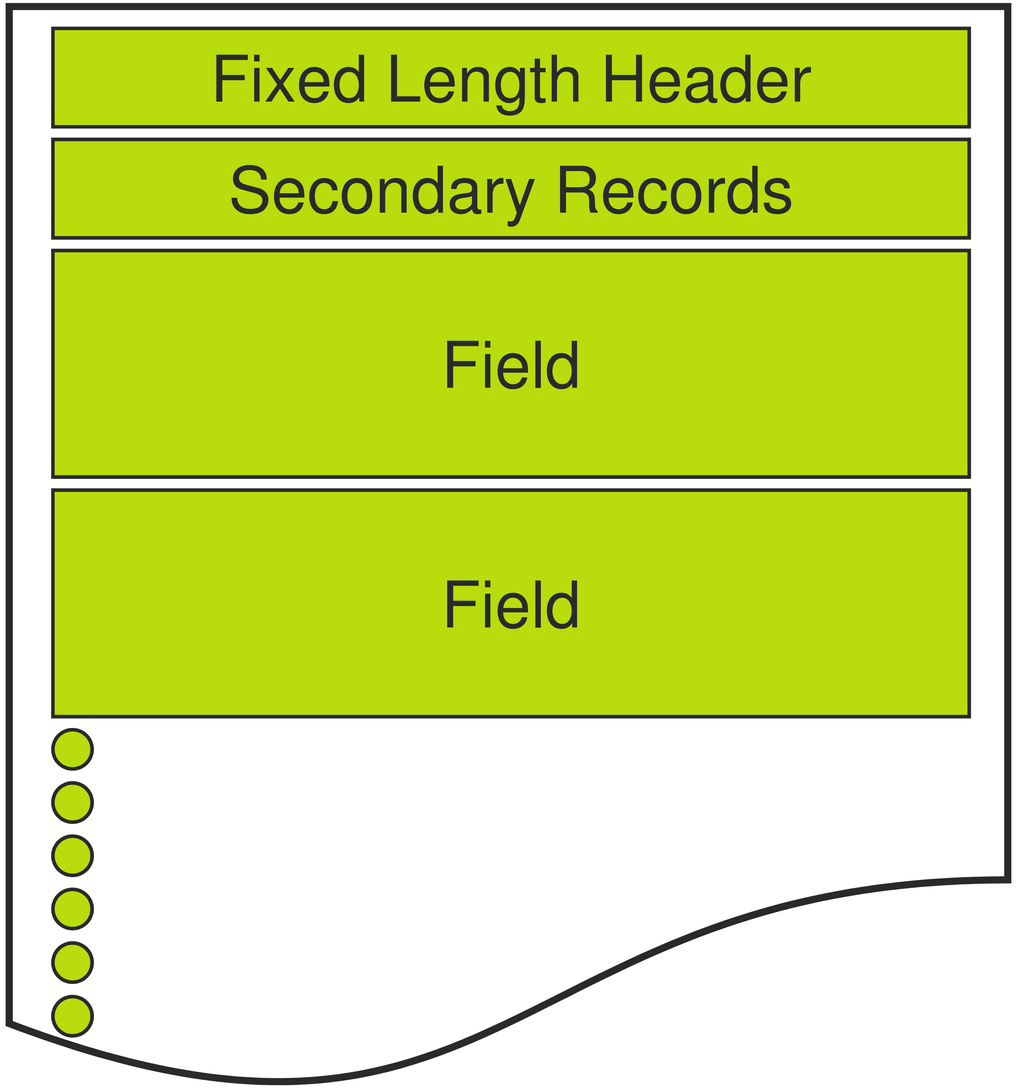
I/O Server (Remote I/O)
- Written in Fortran
- OpenMP + MPI Parallelism
- Multiple “Roles”

Portio (Local I/O)
- Written in C
- Serial Code*
- Multiple “Layers”

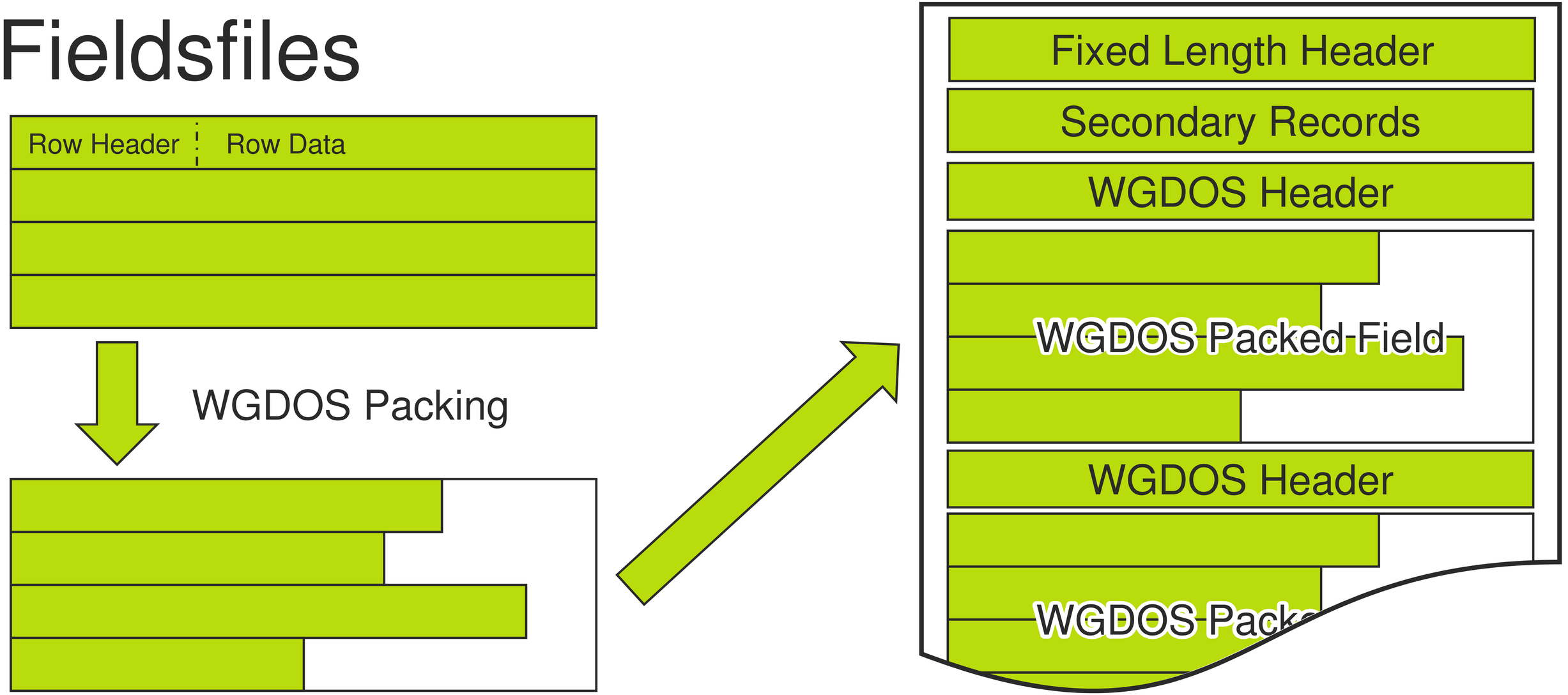
MPI-IO

Fieldsfiles

- Bespoke format (inspired by early GRIB)
- Common “Fixed Length Header”
- Secondary records include constants table, lookup table…
- Fields can be compressed – e.g. WGDOS Packing



Fieldsfiles



NetCDF

- Can also output in NetCDF
  - Not as commonly used
  - Not start dumps
- Some ancillary input uses NetCDF files

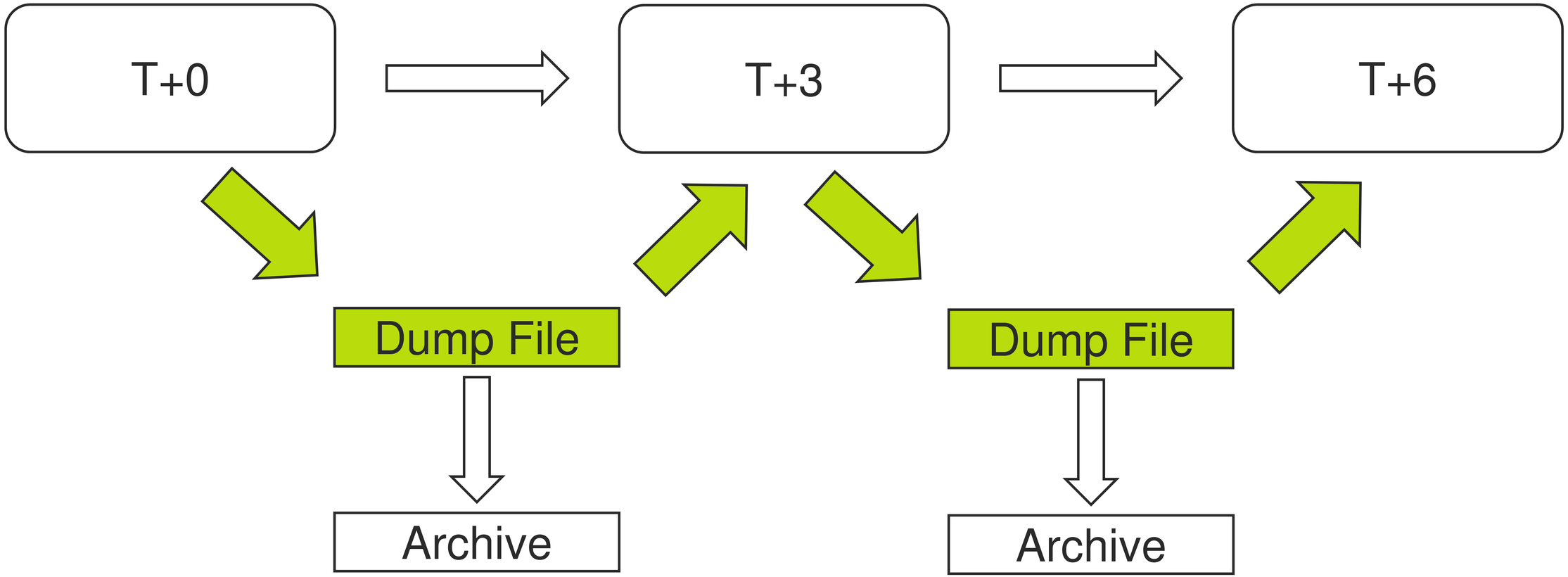
Met Office

Why is I/O a Problem?

Have to deal with:

- Performance
  - I/O can be serial bottleneck
  - Even where we can parallelise, scaling may be poor/limited
  - Significant chunk of runtime
- Cost
  - Archive tapes are expensive!
- Increasing resolution is making this worse

Inter-cycle file use



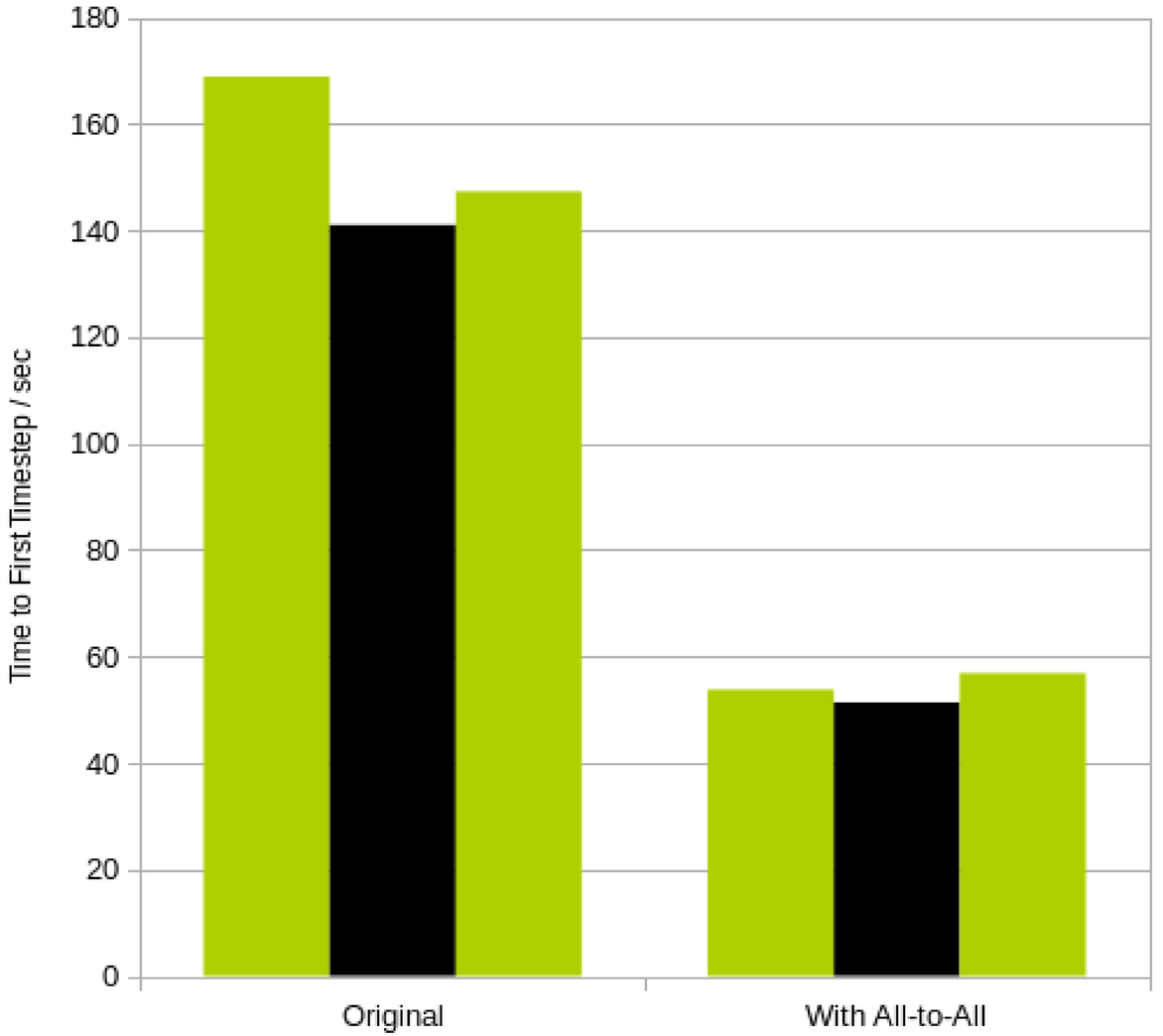
What Have We Done So Far?

Portio Layers

- Internal Buffering → Read Buffering. (+ pre-loading)
- Lustre integration → Support for e.g. striping
- ‘Helper’ threads → Accelerate e.g. buffering
- Add debugging layers → Trace, Time, Throttle, Blackhole

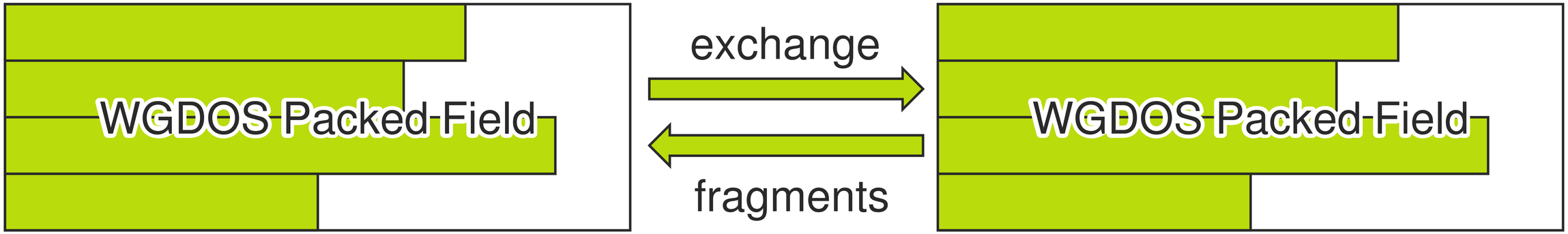
All-to-All Dump Reading

- Read dump fields in parallel; Use MPI_Alltoall to distribute
- Based on work done by Cray
- Example timings shown for three runs of an operational-style n1280 global model.



MPI-IO in IO Server

- Write output in parallel
- Based on work done by Dale Roberts (Australian National University)
- Gather fields onto IO Server ranks; write in parallel (mpi_file_write_at_all)
- Complications from packed fields



Other Optimisations

- Namelists read by only one rank
- Reduction of enquiry functions
- Ensure only one rank opens files in serial case
- Ensure correct compiler flags. (No debug symbols with Cray Compiler)

Current Challenges

I/O Stalls On Lustre

- Usually get good performance
- However, occasionally see a long stall (maybe ~10x I/O time)
- Stalls are infrequent, but not negligibly rare
- Mechanism not properly understood, but…
- Hypothesis that stalls are related to Lustre response
  - When they occur, stall time is consistent.
  - More probable when higher frequency of transactions
    - … when we have more threads
    - … when we have smaller block sizes
  - Episodic

What Do We Still Need To Do?

I/O Server

- Work on queues
  - Want to ensure I/O server is responsive and does not block overlapping computation

Portio

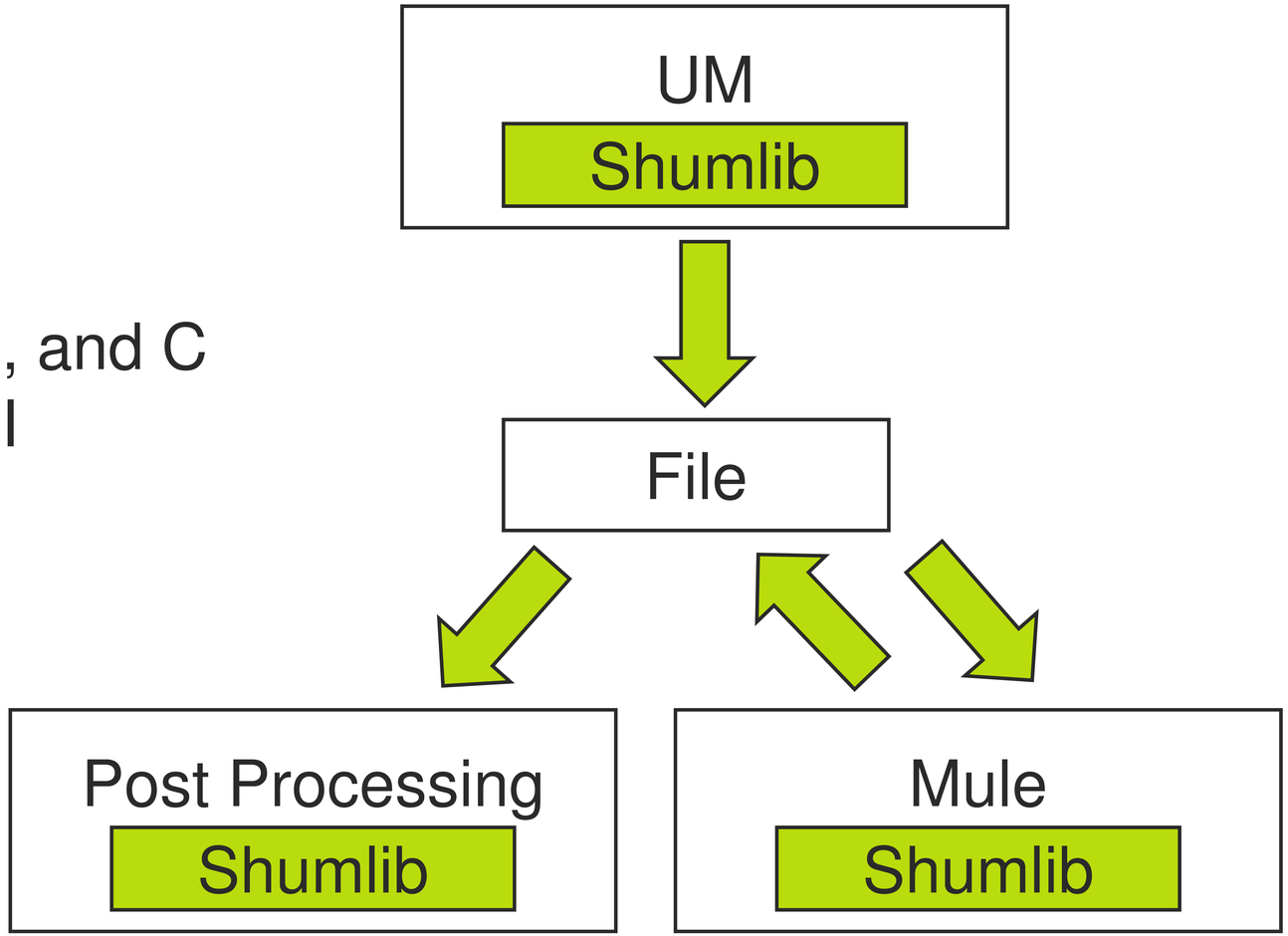
- Enable write helper thread
- Add new layers for other file-system architectures?
- (Migrate to Shumlib)

Other Changes

- Extend All-to-All method to other files – IAU, LBCs
- Alternative compression?

Shumlib

- SHared UM LIBrary
- Provides bindings in Fortran, Python, and C
- High level Fieldsfile manipulation API
- And low level I/O API
- 3-Clause BSD licence
- (Plus other non-IO related code)



Conclusions

Conclusions

- Taken some steps to deal with I/O performance in the UM
- Can now have parallel reads and writes of most important files
  - This is most important for operational runs
- But still need to do more – e.g. can extend to other files
- We still need to do work to understand fully the low level details
  - Why do we get stalls?

Thanks

Any Questions?