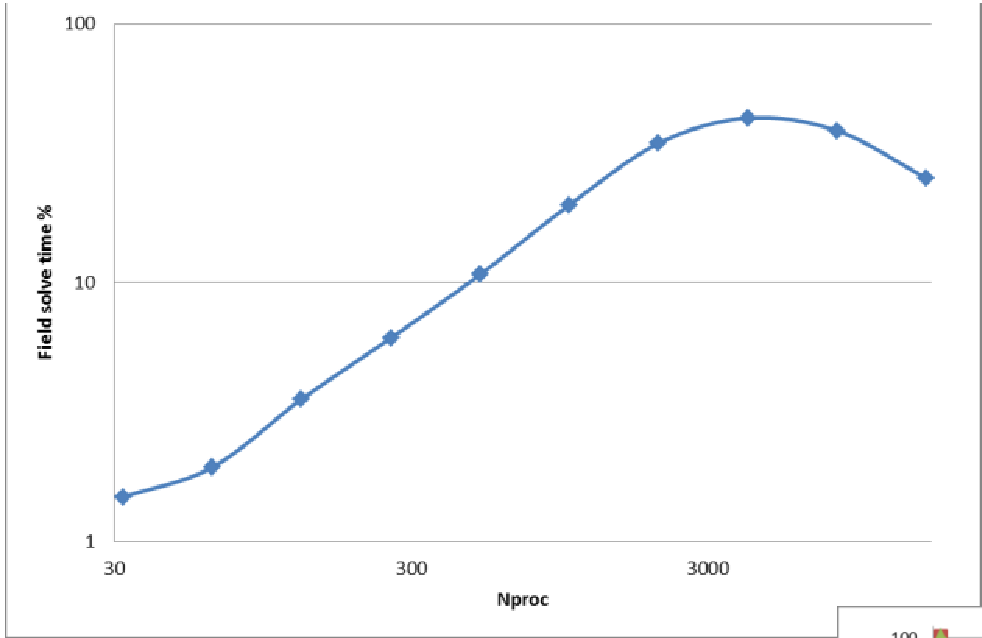
OPTIMISED DATA DECOMPOSITION FOR REDUCED COMMUNICATION COSTS

Manos Farsarakis
Adrian Jackson, EPCC, a.jackson@epcc.ed.ac.uk, @adrianjhpc
David Dickinson, York
Colin Roach, CCFE

epcc

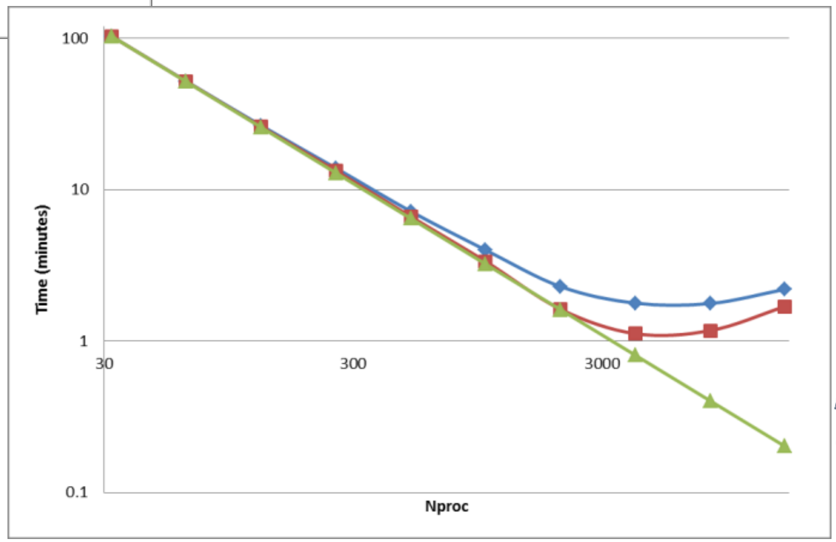
THE UNIVERSITY OF EDINBURGH

Application scaling



- GS2: Flux-tube gyrokinetic code
  - Initial value code, solves the gyrokinetic equations for perturbed distribution functions together with Maxwell's equations for the turbulent electric and magnetic fields
  - Linear (fully implicit) and Non-linear (dealiased pseudo-spectral) terms
  - Different species of charged particles

- The optimised code efficiency is ~80% for the collisional problem at 1024 cores, and 50% at 4096 cores



THE UNIVERSITY OF EDINBURGH

Fields calculation

- Domain decomposition optimised the linear layout
  - Splits spatial domain across processes
  - Requires some communication for global values
- Non-linear and collisions require different layouts
  - Non-linear involves FFT transformation
- Fields calculation requires reverse of linear domain layout
  - Sections of the spatial domains need to be combined
  - Velocity space local
- Each part of the time step requires some communications

Velocity space integration

- Velocity space integration in fields
  - Currently calculation is done as a loop as follows:

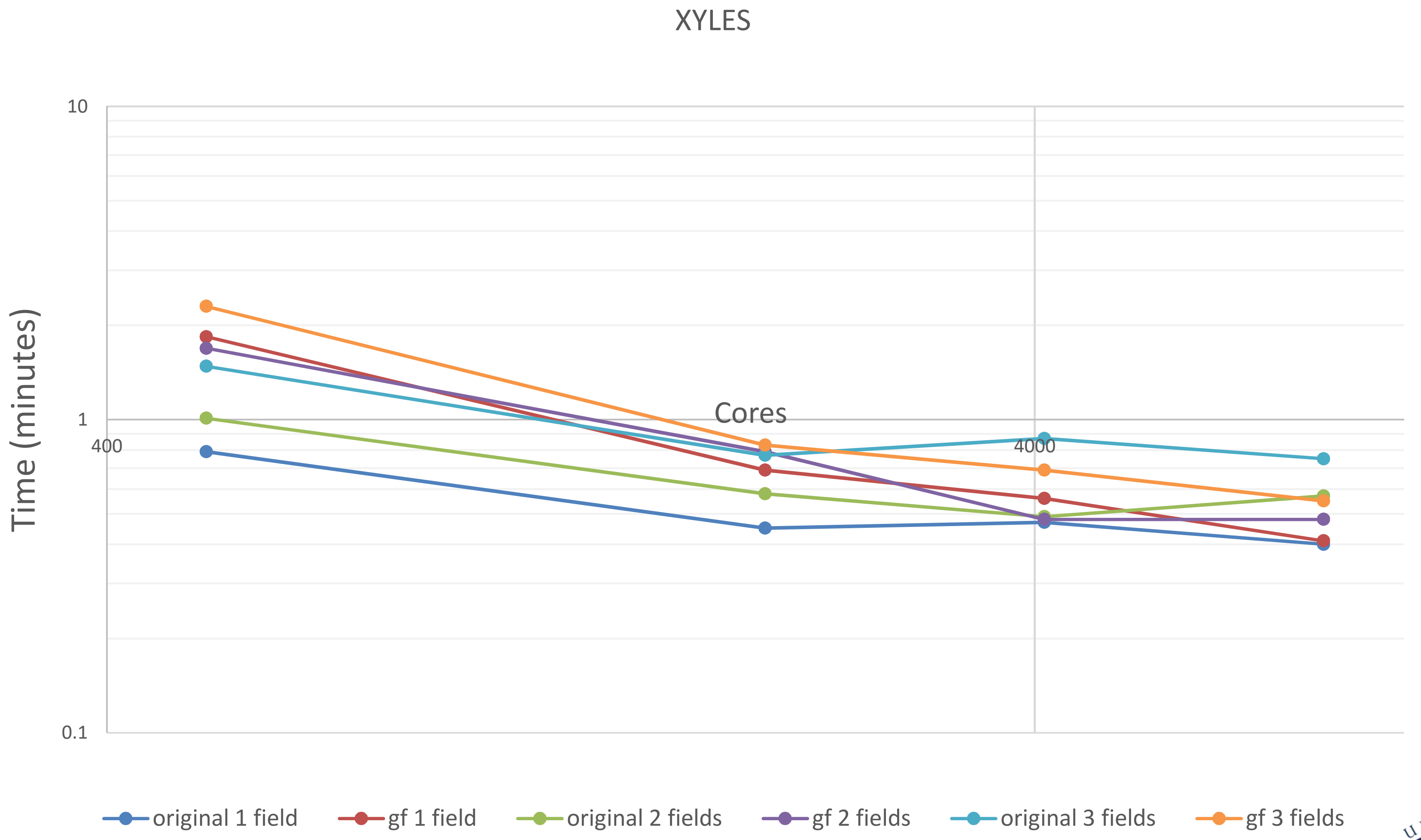
```
do iglo = g_lo%llim, g_lo%ulim
   do naky
      do nx
        Perform calculation
      end do
   end do
end do
MPI_Allreduce to get final result
```

- This has already been optimised
  - Use of sub-communicators to restrict all reduce to processors that share x-y points
- Aim to remove the all reduce completely
  - Perform a data redistribute before integration

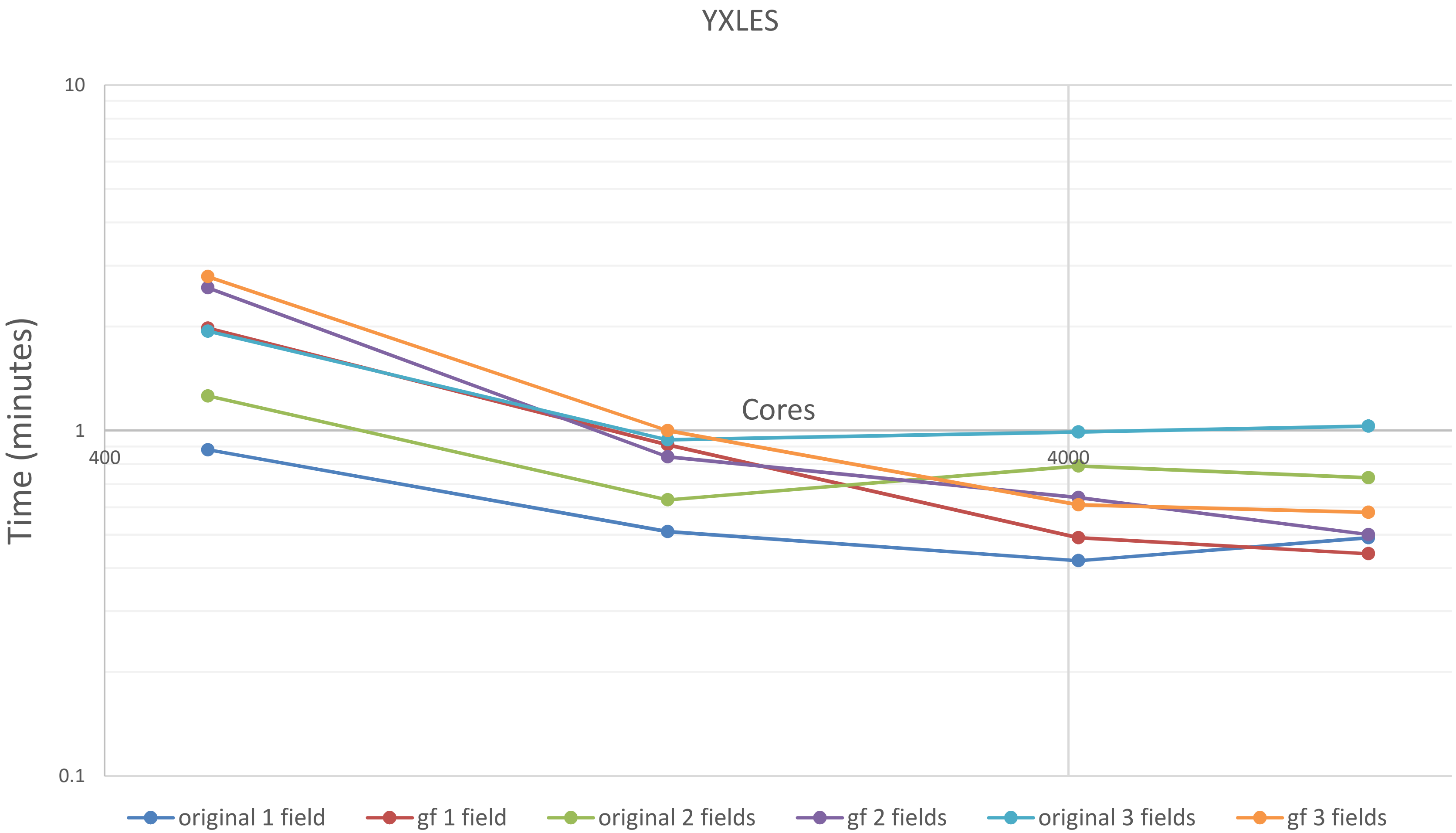
Velocity space integration

- Replace current all reduce with redistribute that does data transpose
  - Send all data from single x-y point to a given processor
  - Perform the integration for that x-y point only on that processor
- Implies some load imbalance at scale
  - x*y is smaller than nproc for large core counts
  - Some processes with zero work for this step
- Create new layout and redistribute object
  - Decompose x-y points to processes
  - Map from linear to fields space
  - Perform the velocity space integration
- Two different decomposition methods
  - Basic rank based assignment
    - First m processes get a gf_lo point
  - Distributed assignment
    - Try to spread gf_lo points out amongst processes

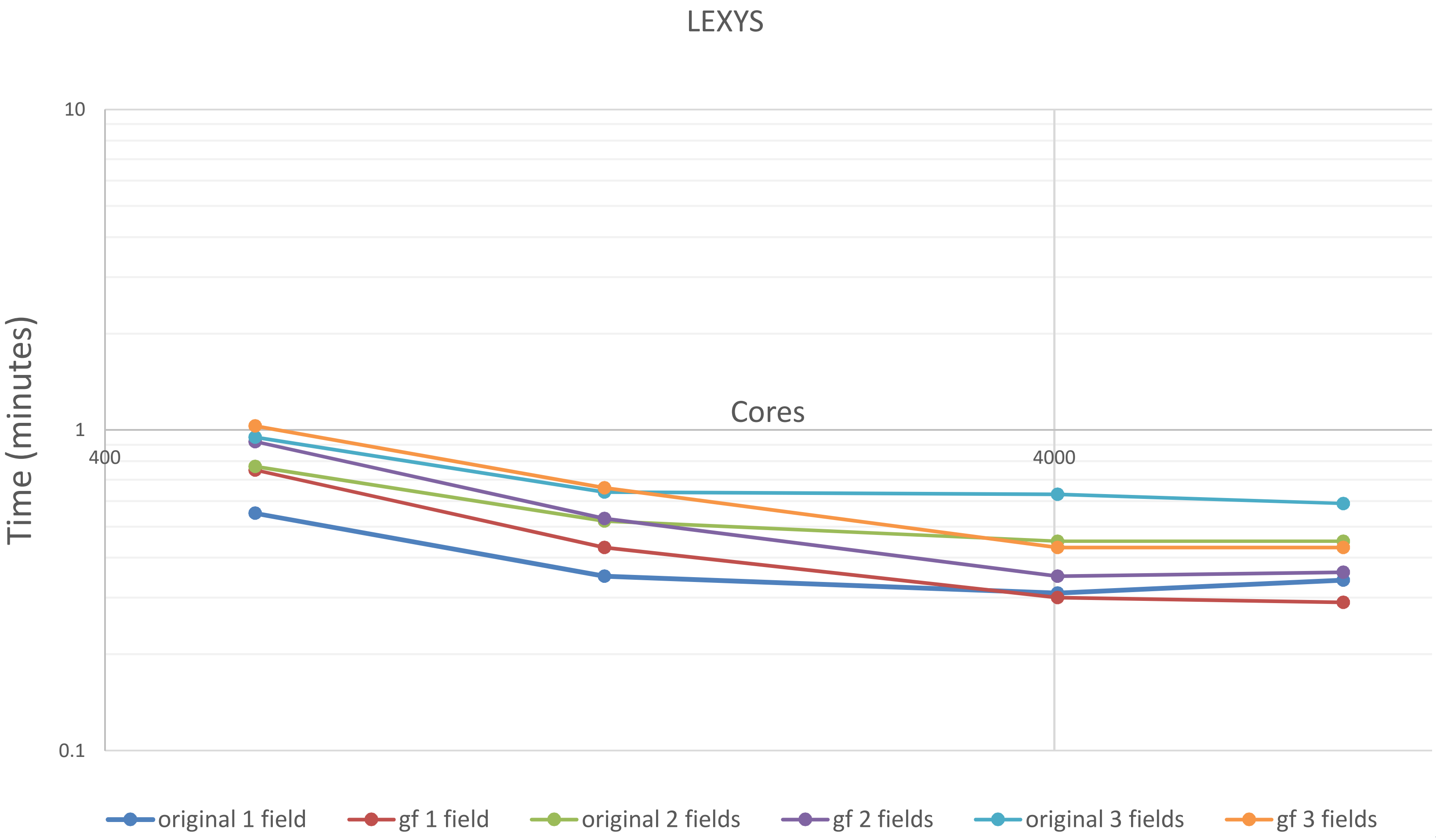
Performance – Advanced Time



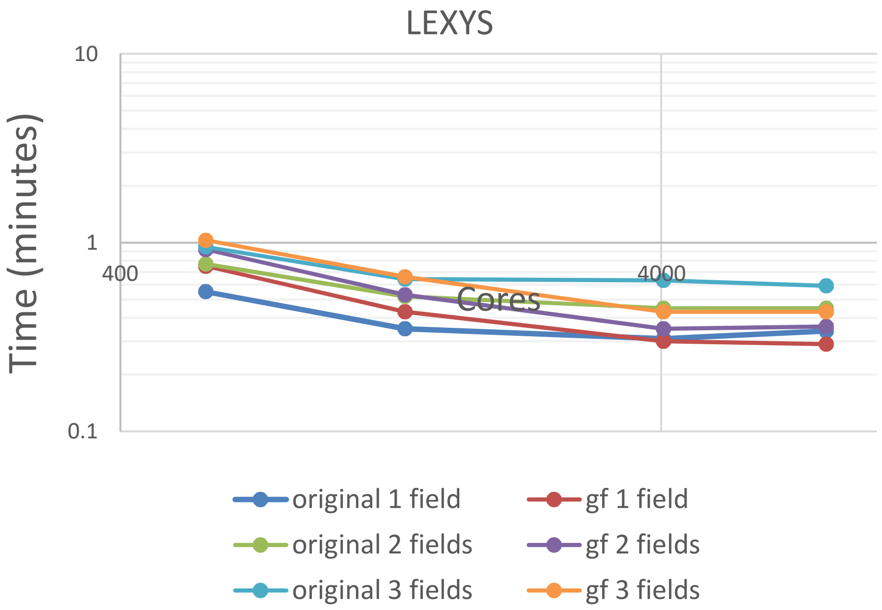
Performance – Advanced Time

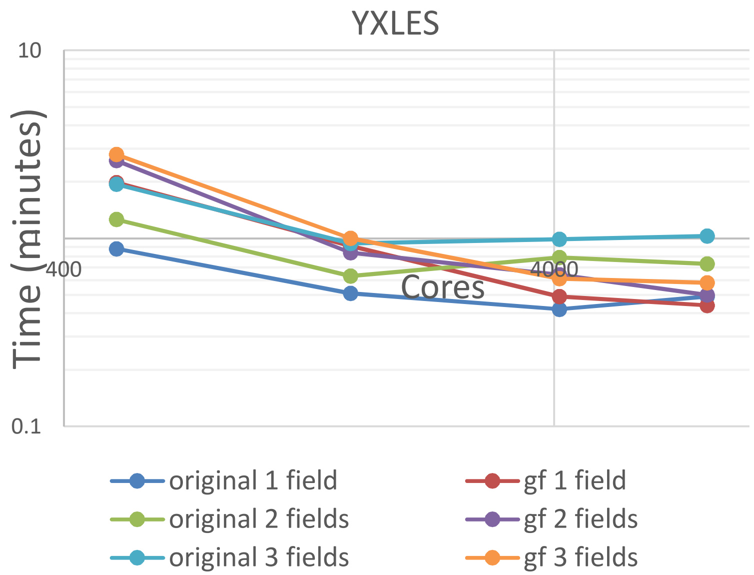


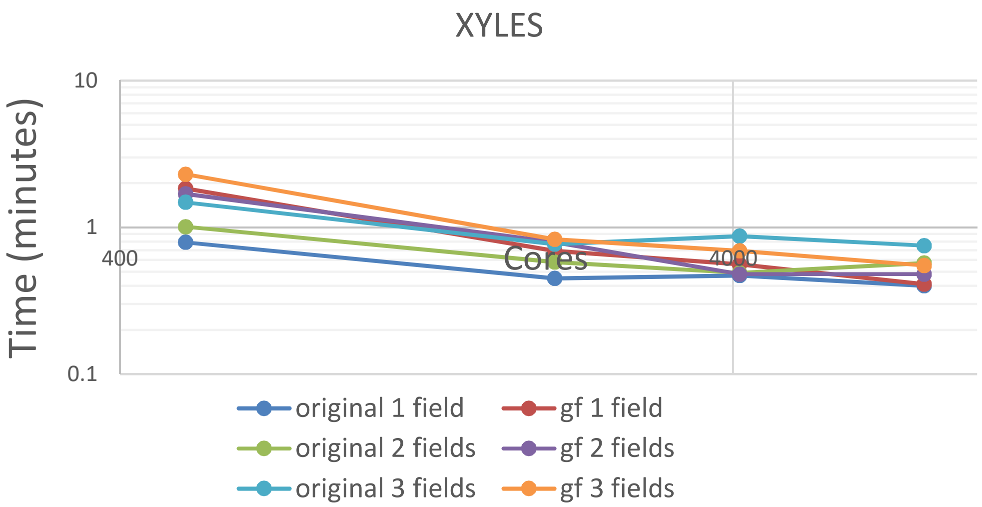
Performance – Advanced Time



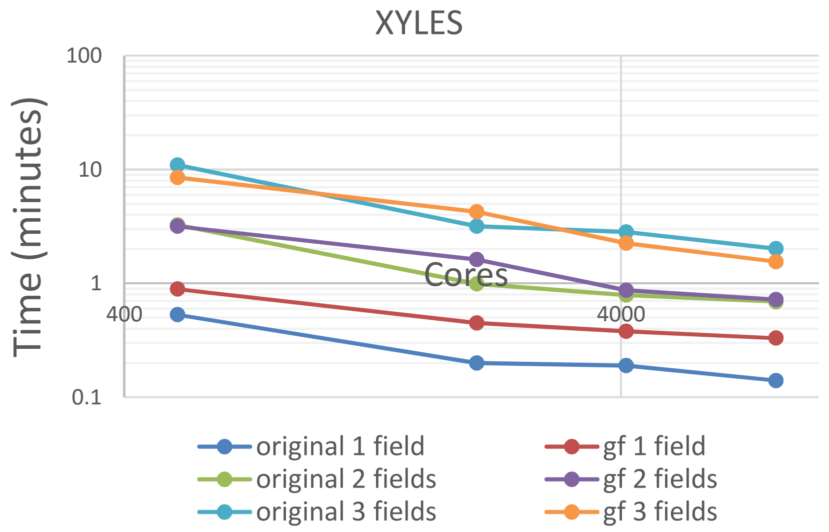
Advanced times

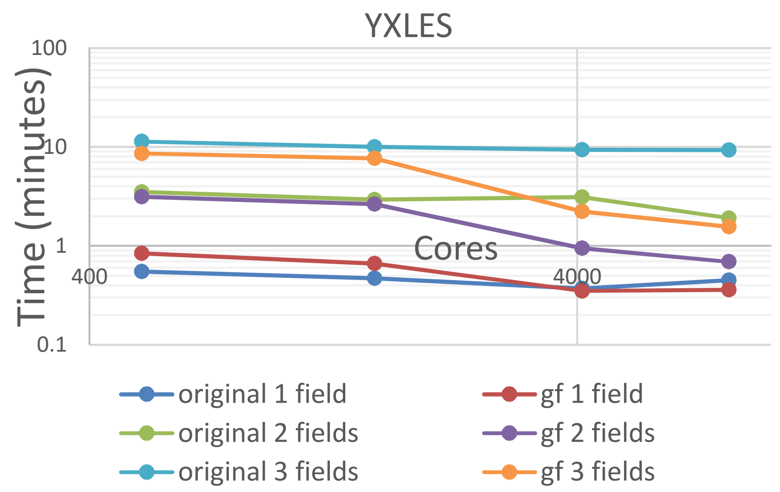


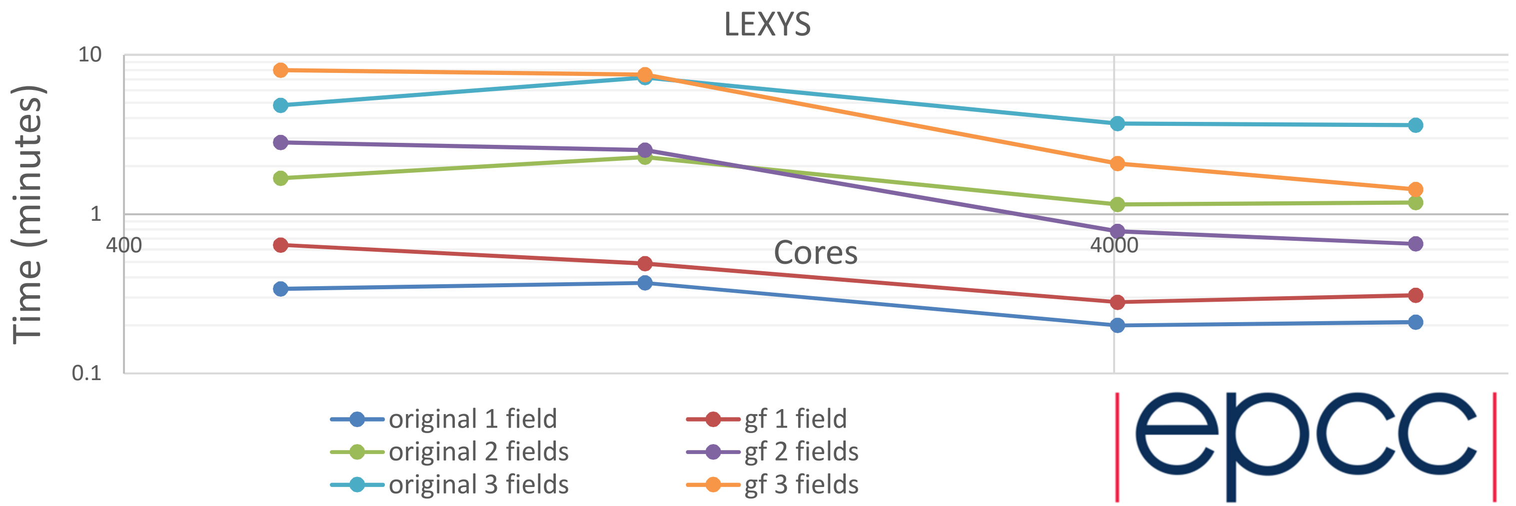




Initialisation times







Summary

- Decomposition data for fields calculation to make computation local improves performance
  - up to 2x performance improvement (depending on number of fields, process count and linear layout)
- The largest performance improvement is for 3 fields and large process counts
- Domain decomposition is inherently load imbalanced
  - A subset of processes have no work to do
  - This subset grows as simulations are strong scaled
    - i.e. for the simulations shown here, there are only 1008 field points, so at 4032 processes, 3024 (75%) will be idle during the fields calculation
- Communication patterns changed
  - Reduces collective communications
  - Increases point to point communications

Full details and future work

- Future work
  - Currently the mapping of the fields decomposition to processes is simplistic:
    - first N MPI ranks get the field points
  - This could be optimised by choosing a mapping that reduces data movement from processes
    - i.e. considers current data locality
  - This could be optimised by choosing a mapping that reduces data movement between nodes
    - i.e. optimise for fast on-node communiations where possible
- Technical report:
  - http://www.archer.ac.uk/community/eCSE/eCSE02-08/eCSE02-08.php

- **Funded by EPSRC ARCHER eCSE program**
- **Funded by Intel IPCC collaboration with EPCC**

epcc

THE UNIVERSITY OF EDINBURGH