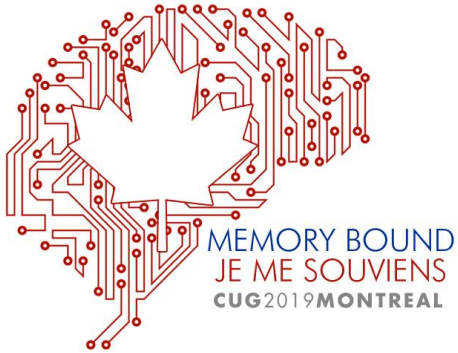
Modelling the earth's geomagnetic environment on Cray machines using PETSc and SLEPc



Nick Brown, Data Architect at EPCC

n.brown@epcc.ed.ac.uk

epcc

THE UNIVERSITY OF EDINBURGH

Background

British Geological Survey

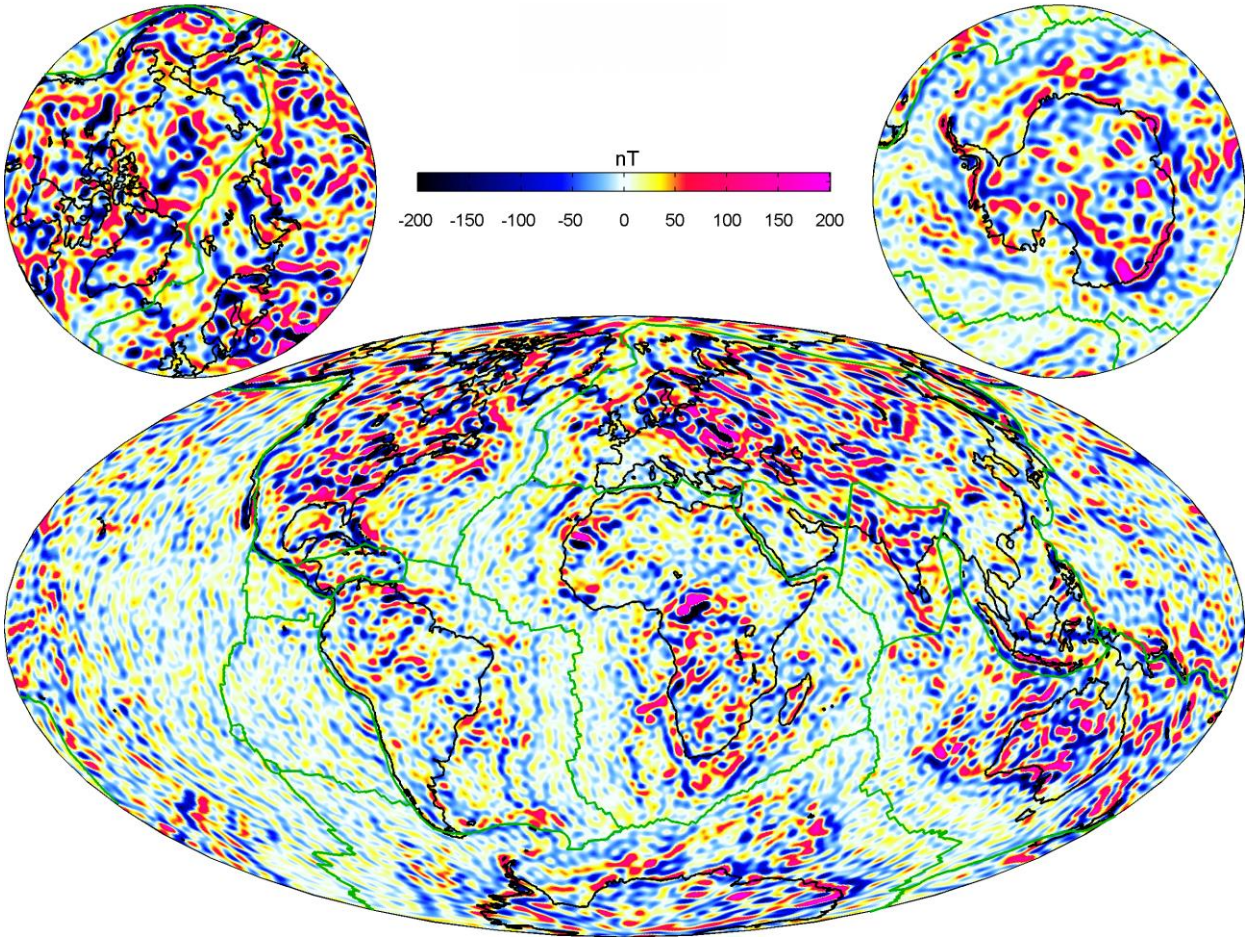
- The Model of the Earth Magnetic Environment (MEME) captures time-varying and the spatially-varying components of the magnetic field.
  - Developed by BGS

- Takes input from
  - Ørsted, CHAMP and ESA swarm satellites
  - Ground based observatories
- Output used for
  - Scientific study & new geophysical understanding
  - BGS Global Geomagnetic Model (BGGM)
  - World Magnetic Model (WMM)

epcc

The University of Edinburgh

Background



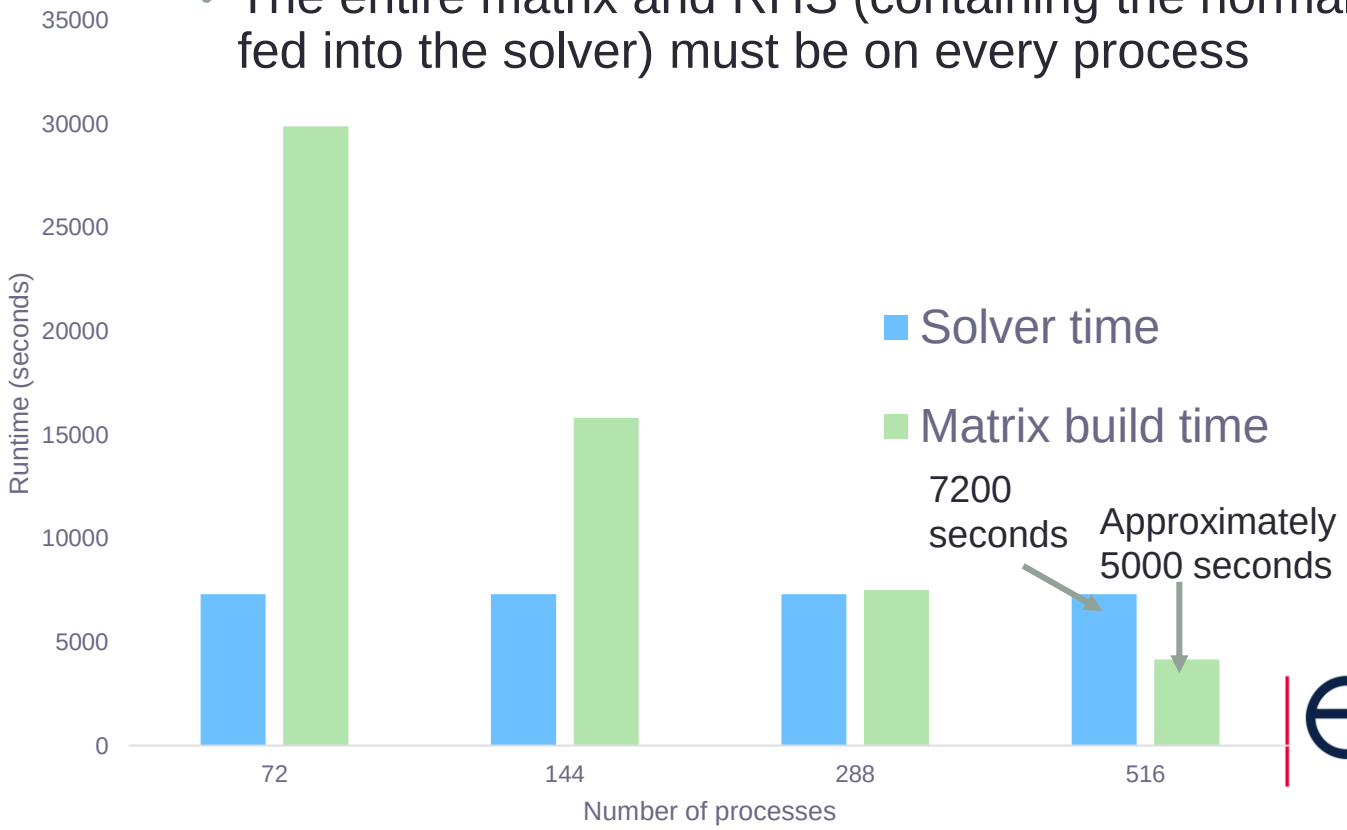
- A model of the vertical component of the lithospheric field
  - Earth's lithosphere includes the crust and the uppermost mantle (hard and rigid outer layer of the earth.)

epcc

THE UNIVERSITY OF EDINBURGH

Existing model performance

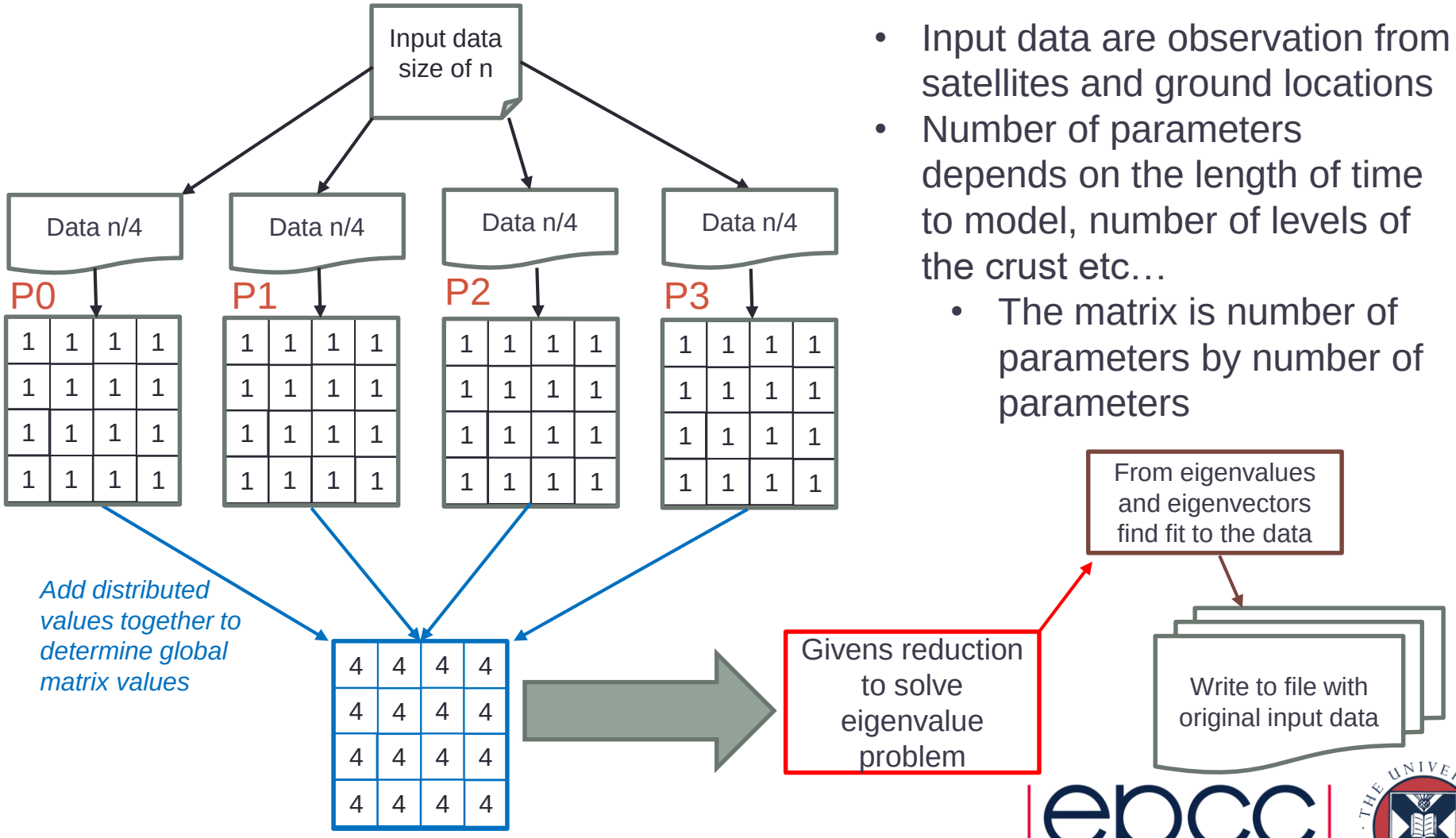
- Benchmarking run on ARCHER, Cray XC30, with 8000 parameters and 4 million data points
  - Solver is sequential
  - Parallelism limited to around 500 processes
  - The entire matrix and RHS (containing the normal equations then fed into the solver) must be on every process



epcc

THE UNIVERSITY OF EDINBURGH

Existing model approach



- Input data are observation from satellites and ground locations
- Number of parameters depends on the length of time to model, number of levels of the crust etc…
  - The matrix is number of parameters by number of parameters

Why PETSc/SLEPc?

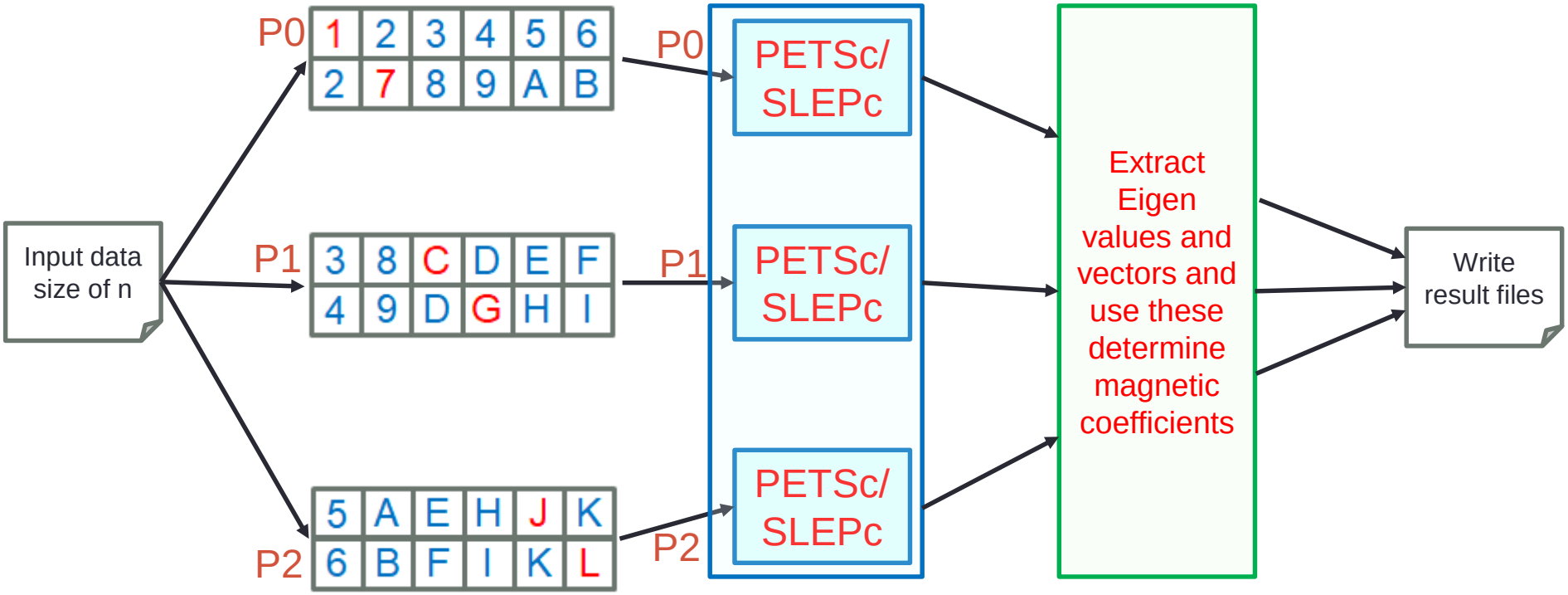
Portable, Extensible Toolkit for Scientific Computation (PETSc)
Scalable Library for Eigenvalue Problem Computations (SLEPc)

- PETSc already part of the Cray module environment ✓
- PETSc has good support for parallelism ✓
- PETSc/SLEPc has good support for Fortran ✓
- Can easily take advantage of PETSc functionality ✓
- Seemed like results would agree ✓

- But
  - SLEPc is best suited to sparse systems, and our system here is quite dense

epcc | THE UNIVERSITY OF EDINBURGH

The plan

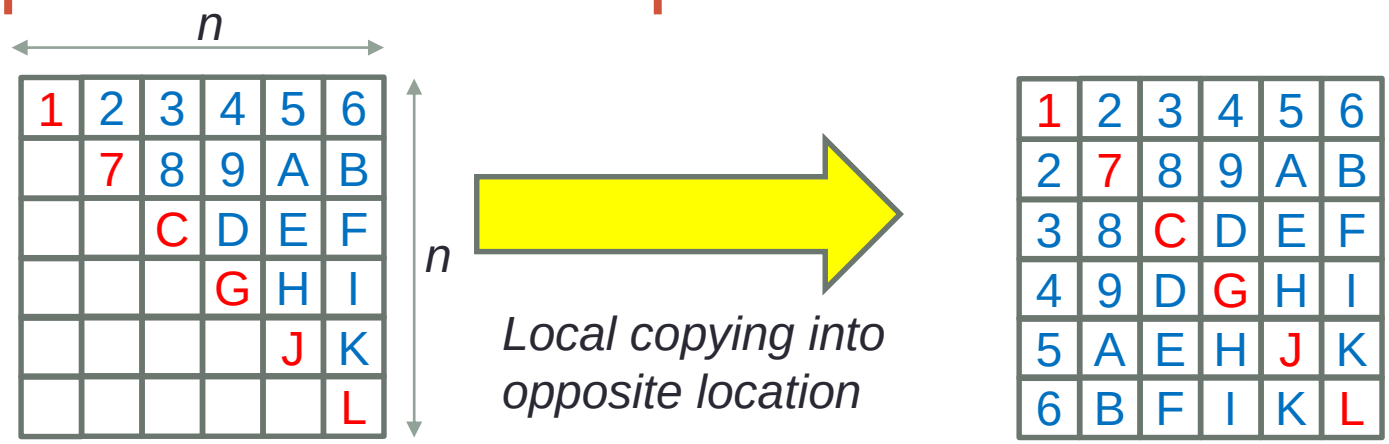


1. Use PETSc and SLEPc to perform Eigen solve
2. Decompose on matrix of normal equations instead of the input data

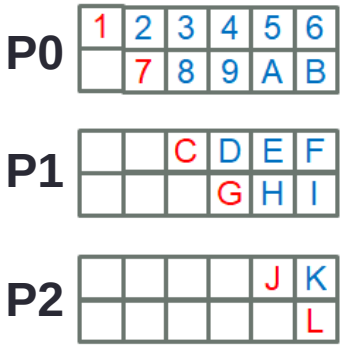
epcc

THE UNIVERSITY OF EDINBURGH

Sequential normal equation matrix building



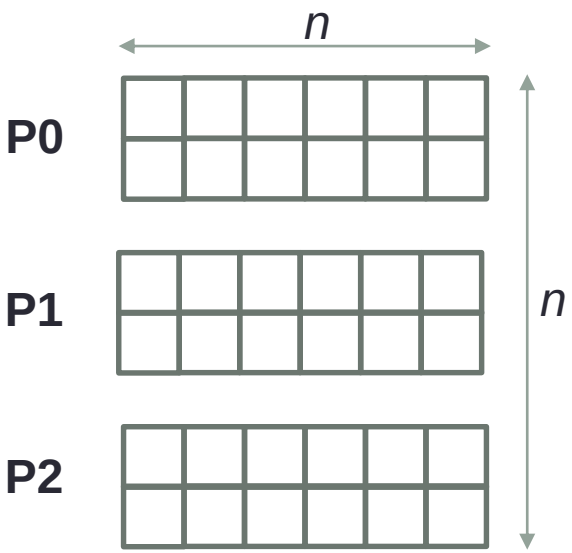
- With entirety of matrix on each process, the symmetry is very simple to deal with as only compute the diagonal and upper part, then copy upper elements into their corresponding lower element locations
  - When we split the matrix up could just calculate upper elements and communicate to the lower elements
    - But significant load imbalance!
  - Or could compute all elements, but duplication of work



epcc

THE UNIVERSITY OF EDINBURGH

Distributed normal equation matrix building



- Total number of points to be explicitly calculated

$$f = \frac{n^2 - n}{2} + n$$

- Base points per row to be explicitly calculated

$$r = \frac{f}{n}$$

f=21

r=3.5

1	2	3	4		
	7	8	9		

		C	D	E	F
			G	H	I

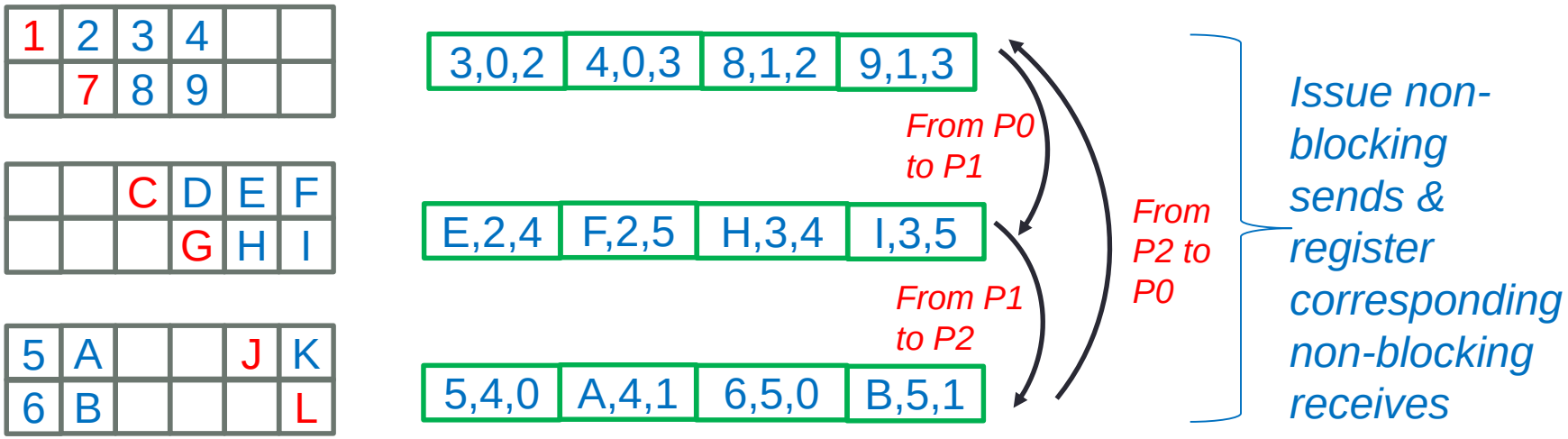
5	A			J	K
6	B				L

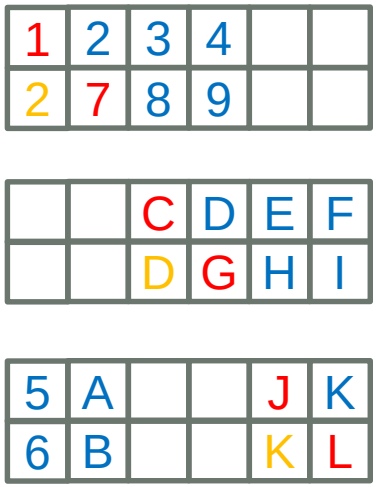
- Starting at the diagonal, start calculating r local points.
  - If r is fractional (n is even), alternate between ceil(r) and floor(r) points for each row
  - If the number of rows/2 is even, then in the second half of the matrix swap over ceil/floor

epcc | THE UNIVERSITY OF EDINBURGH

Distributed matrix building

Each entry is the value as well as the global row and column (16 bytes per entry)





- Next we copy all local values (between locally held rows)
- Once we have done this wait for all communications to complete
  - Overlapping the local data copy with the communications

epcc

THE UNIVERSITY OF EDINBURGH

Distributed matrix building

Received by P0 from P2

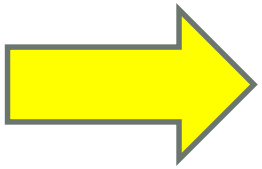
5,4,0	A,4,1	6,5,0	B,5,1

Received by P1 from P0

3,0,2	4,0,3	8,1,2	9,1,3

Received by P2 from P1

E,2,4	F,2,5	H,3,4	I,3,5



Write data into the appropriate place

1	2	3	4	5	6
2	7	8	9	A	B

3	8	C	D	E	F
4	9	D	G	H	I

5	A	E	H	J	K
6	B	F	I	K	L

As each received data value also has associated its global row and column, it is trivial to place it in the appropriate location

- Whilst we still need communication of the values, we don't need communication to coordinate which process calculates what
  - At worst each process needs to communicate with every other process, but this is 1 single large message

epcc

THE UNIVERSITY OF EDINBURGH

Irregular & indirect memory accesses

```
do j=1, n
    do i=1, n
        matrix(dataloc(i), j)=equations(inputdata(i,j)) + ....
    end do
end do
```

- 35% of the runtime when building the normal equations is in a procedure where we have this indirect memory access

Counter description	Value
Number of cycles	69,109,605,287
Number of instructions issued	30,451,871,184
Total resource stalls	60,734,999,355
Resource stalls (store buffer)	60,042,957,250
Cycles no instructions are issued	54,243,641,893
L1 cache hits	5,613,708,007
Instructions per cycle	0.44
Cycles per instruction	2.26
Total % cycles stalled	78

  - This is a potential problem as the memory access pattern is highly irregular and-so it is impossible for the cache, bringing in lines, to benefit here.
  - Unpredictability also makes it impossible for HW prefetcher

epcc | THE UNIVERSITY OF EDINBURGH

Irregular memory accesses

- Used software pipelining and software prefetching to fetch the data ahead of time in a non-blocking fashion

```
do j=1, n
    do i=1, n
        k=i+PREFETCH_DISTANCE
        if (k .le. n) then
            call do_prefetch(matrix(dataloc(k), j))
            call do_prefetch(equations(inputdata(k,j)))
        end if
        matrix(dataloc(i), j)=equations(inputdata(i,j)) + ....
    end do
end do
```

Counter description	Value
Number of cycles	32,834,830,469
Number of instructions	83,862,040,343
Total resource stalls	1,903,905,993
Resource stalls (store buffer)	629,838,931
Cycles no instructions are issued	1,927,696,028
L1 cache hits	19,907,028,904
Instructions per cycle	2.53
Cycles per instruction	0.39
Total % cycles stalled	5.8

With software pipelining & prefetching

Counter description	Value
Number of cycles	69,109,605,287
Number of instructions issued	30,451,871,184
Total resource stalls	60,734,999,355
Resource stalls (store buffer)	60,042,957,250
Cycles no instructions are issued	54,243,641,893
L1 cache hits	5,613,708,007
Instructions per cycle	0.44
Cycles per instruction	2.26
Total % cycles stalled	78

Original procedure, no prefetching

- Over 3 times the number of instructions, but reduces overall runtime of this procedure by almost three times

epcc | THE UNIVERSITY OF EDINBURGH

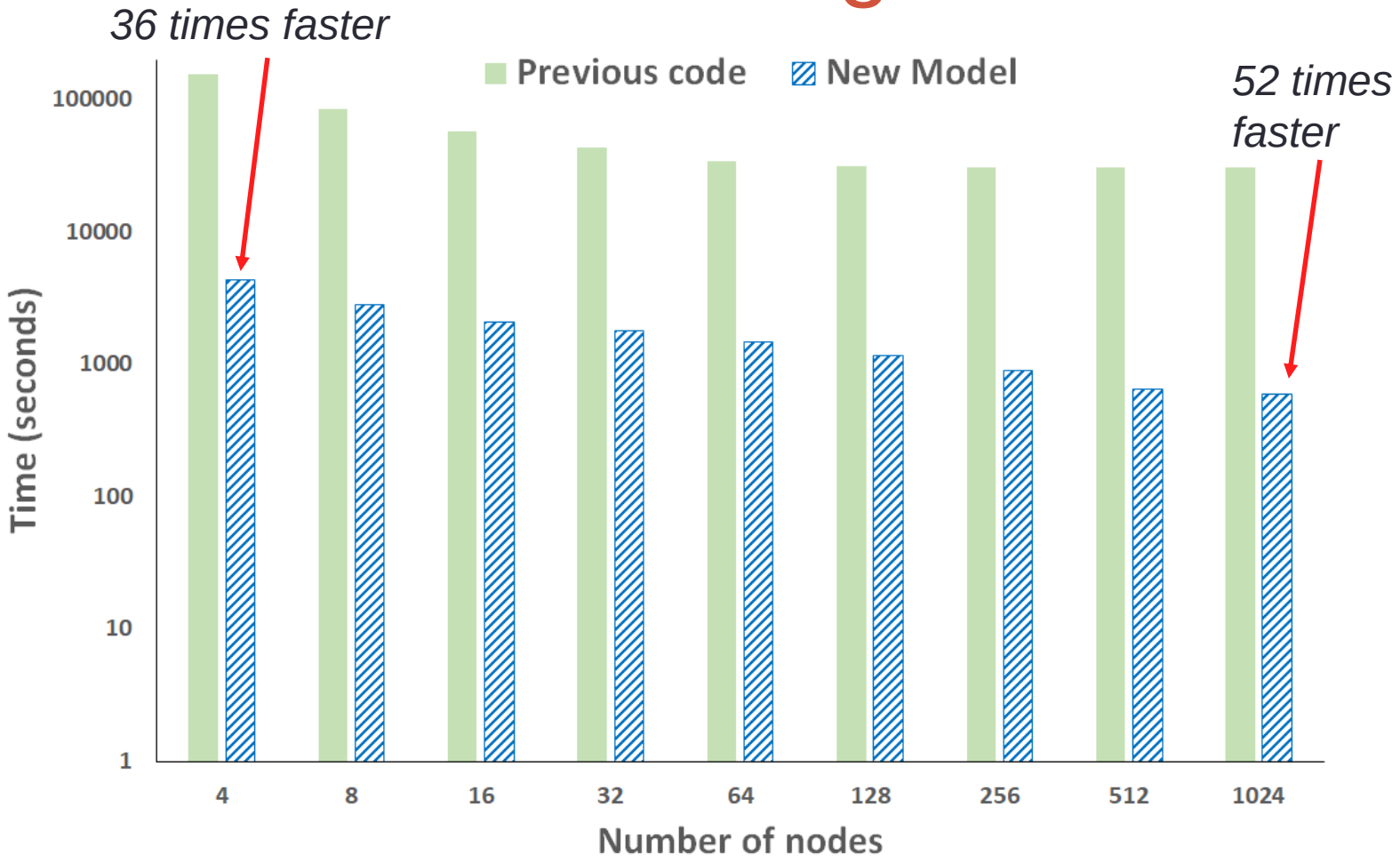
Integration with SLEPc

- Once the normal equations are built, actually integrating the SLEPc solver into the code is fairly easy.
- Search for all eigenpairs on the normal equations matrix and then apply the eigenvalue and eigenvector to the RHS with some weighting.
  - Experimented with all the different solvers and options, found the default Krylov-Schur gives the best performance, both in terms of result accuracy and runtime
  - We found it important to tell SLEPc our matrix is Hermitian

epcc

THE UNIVERSITY OF EDINBURGH

Performance and scaling



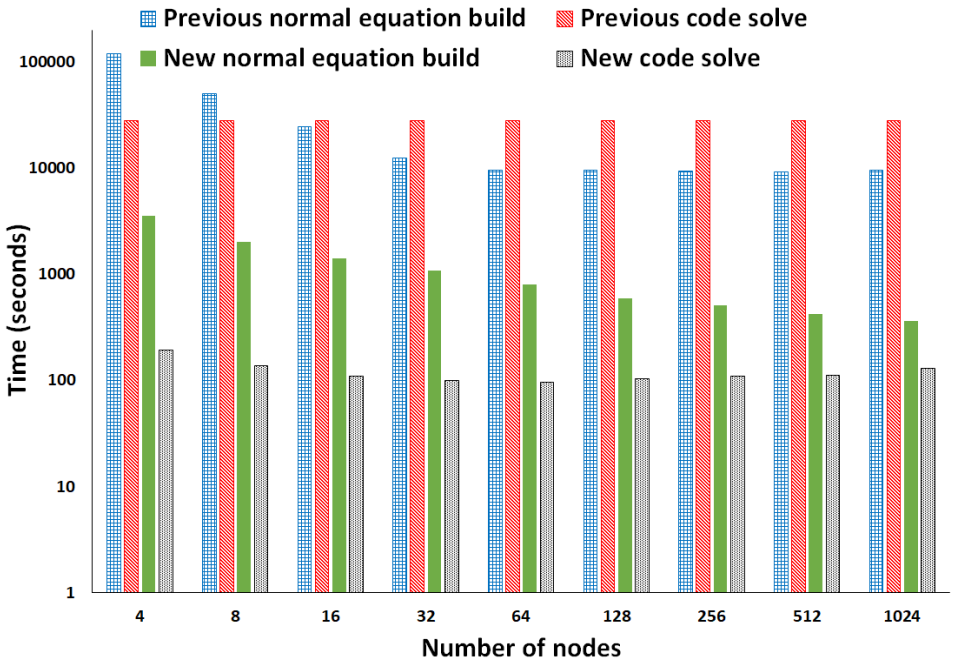
- 10,000 model coefficients with 4.8 million items of data on ARCHER, Cray XC30 using Cray compiler version 8.6.5, PETSc version 3.8.4 and SLEPc version 3.8.3.

epcc

THE UNIVERSITY OF EDINBURGH

Performance and scaling

- For the solver the new model takes 194 seconds over 4 nodes and 97 seconds over 64 nodes in contrast to 28,500 seconds for the previous model!
  - Building of the normal equations is also significantly faster, but this is the largest gain



Metric	Minimum %difference	Maximum %difference	Mean %difference
Ffit	0.000019	0.000053	0.000026
Fobsfit	0.000014	0.000029	0.000019
XYZfit(1)	0.000010	0.034425	0.000091
XYZfit(3)	0.000010	0.312869	0.000204
XYZobsfit(1)	0.000010	0.000081	0.000033
XYZobsfit(3)	0.000010	0.182673	0.000130
XYZfit_c(1)	0.000010	0.007966	0.000073
XYZfit_c(3)	0.000010	0.011187	0.000085
Model coefficients	0	43	0.048114

- Using SLEPc our results still match closely with the previous Givens reduction approach

epcc

THE UNIVERSITY OF EDINBURGH

Scaling beyond 10,000 parameters: Memory usage challenges

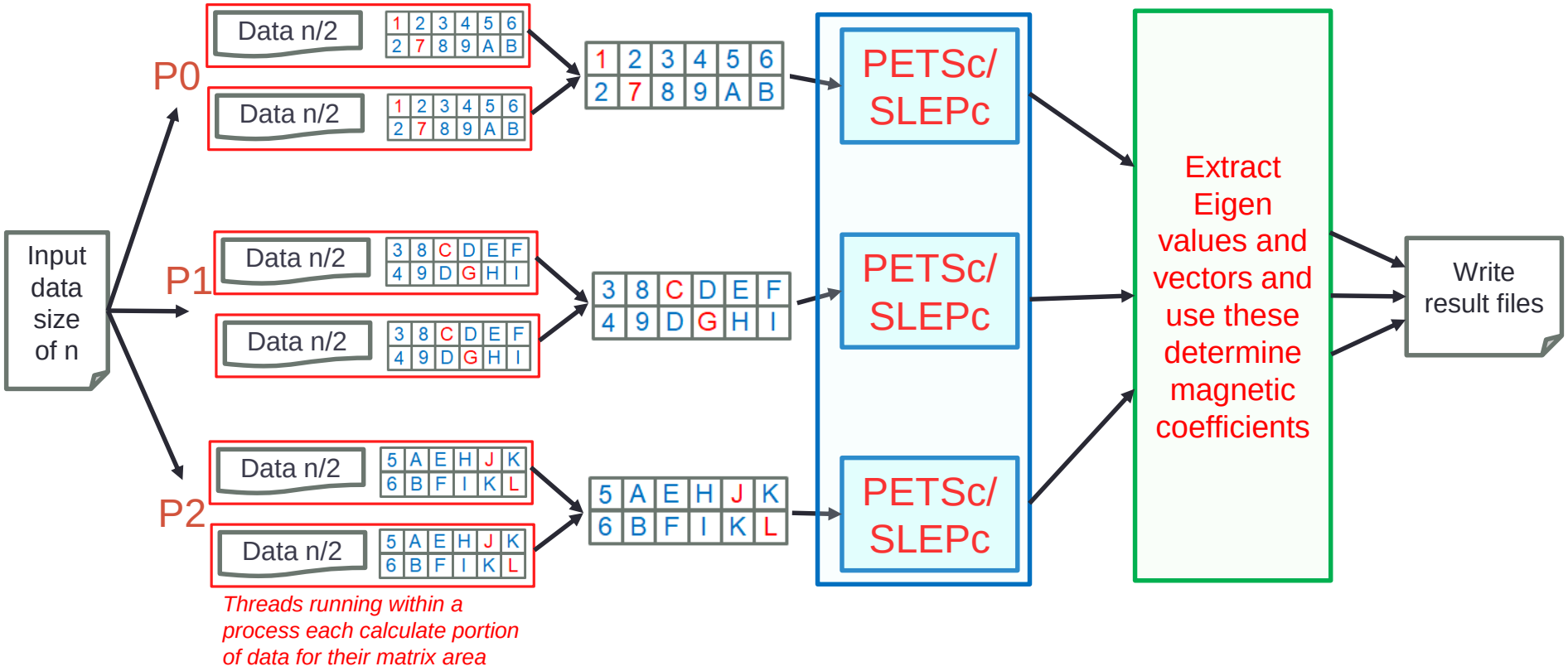
- When scaling beyond the 10,000 coefficients limit of the current model, we start seeing non parallelisable memory limitations in SLEPc.

Number of coefficients	Direct solve size (MB)
10000	1498
20000	9543
30000	23908
40000	38200
50000	57300
60000	85950
70000	114600

- There are very many options in SLEPc, but from detailed experimentation with all of these we were unable to avoid the very high memory overhead (per process)
  - This is because we are working with a very large matrix and searching for all eigenpairs in this matrix, so it's a hard problem to solve

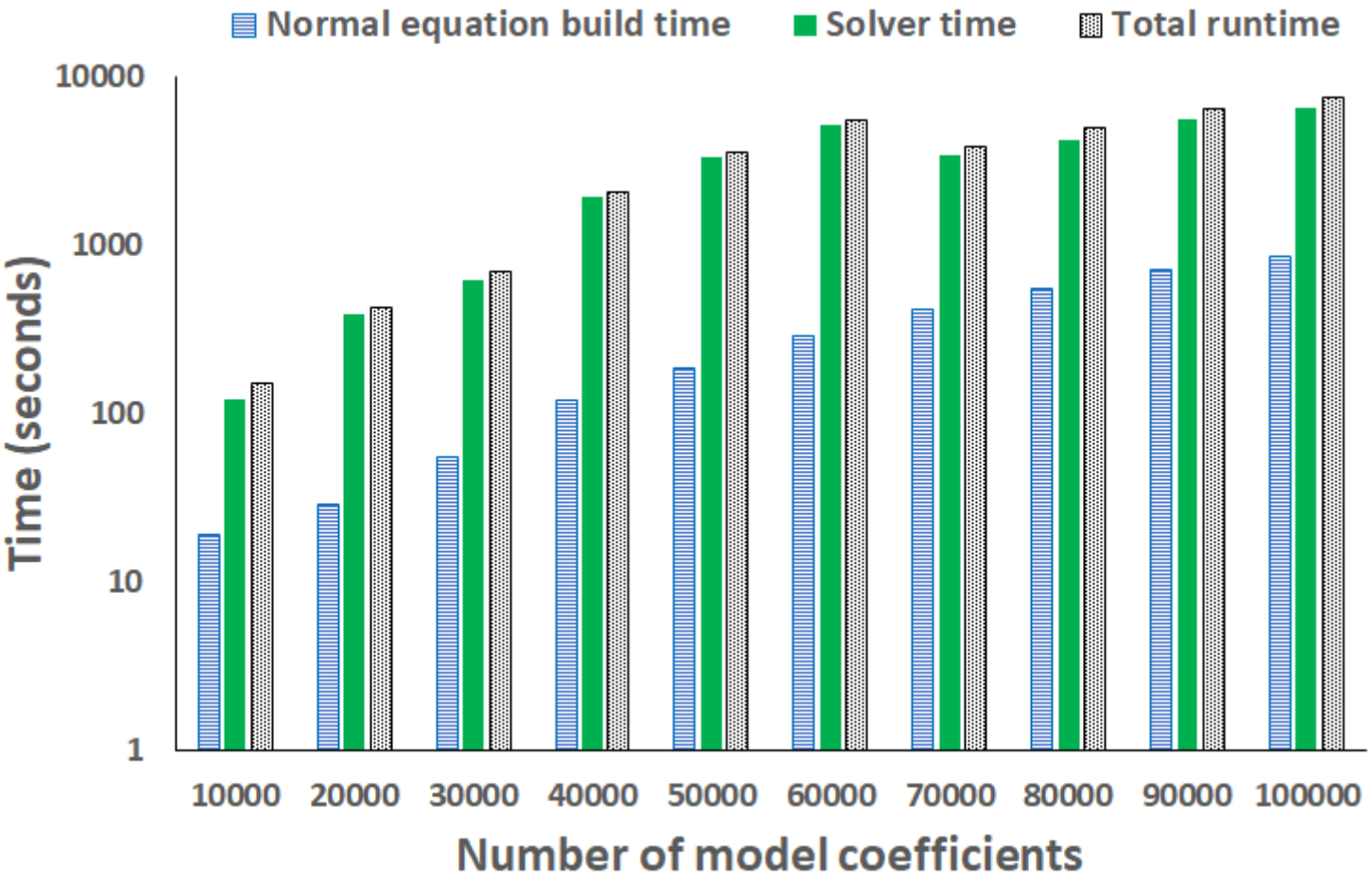
epcc | THE UNIVERSITY OF EDINBURGH

Hybridising the model using MPI/OpenMP



- This memory requirement is on a process by process basis, hence we aim to reduce the number of processes and use OpenMP to thread over cores

Large scaling of the new model



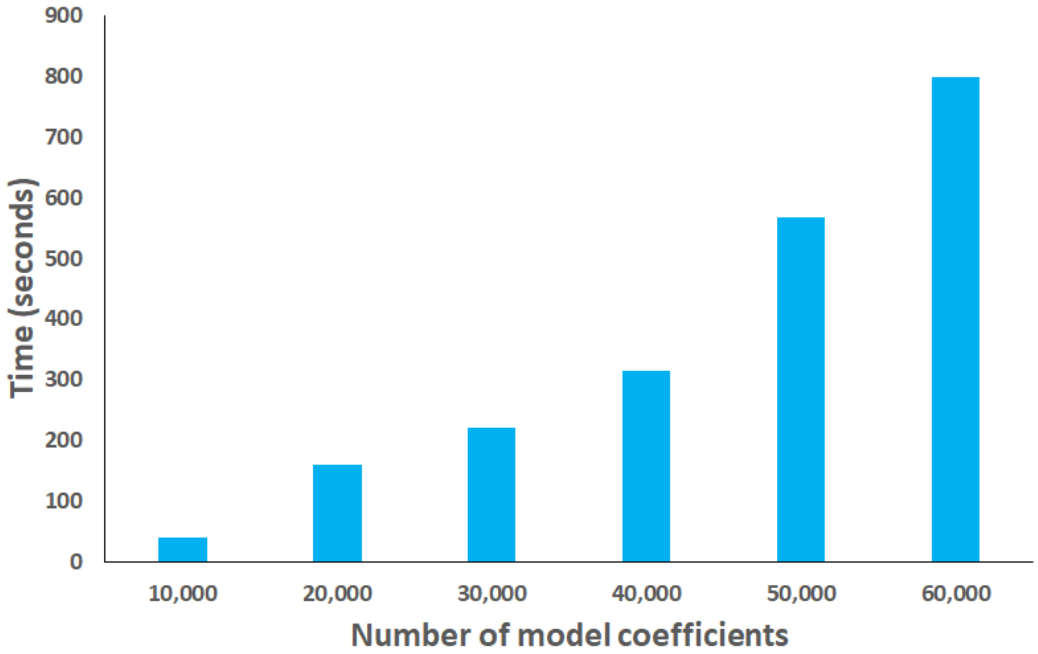
- On 40 nodes of ARCHER, Cray XC30 using Cray compiler version 8.6.5, PETSc version 3.8.4 and SLEPc version 3.8.3.

epcc

THE UNIVERSITY OF EDINBURGH

Iterative vs direct solver

- So far the scientists have replied on a direct Eigen solver, but what about an iterative solver?
  - Replace the SLEPc direct solver with an iterative one from PETSc (e.g. GMRES.)
  - From a code perspective this is trivial, and that's a major benefit of SLEPc
  - Ran experiment on 40 nodes of ARCHER, Cray XC30

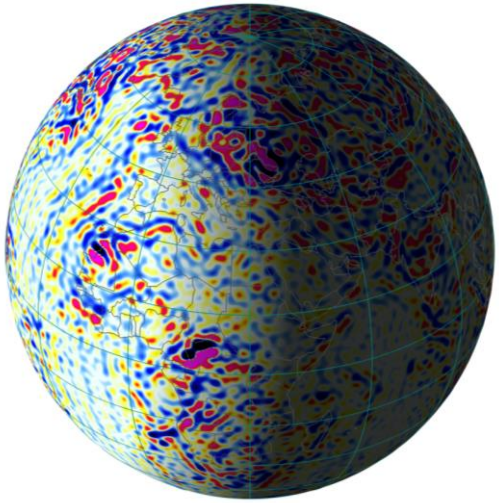


- Fixes the memory issue and the performance is significantly better
- But not fully stable and less so as we increased the problem size

epcc

THE UNIVERSITY OF EDINBURGH

Conclusions



- PETSc and SLEPc are a good combination
  - Lots of options and trivial to change between them
  - Possible to get great performance, especially if not after all the Eigenvalues
  - Memory is an issue with the direct solver
- Building of the normal equations required some thought
  - As it is a symmetric matrix had to consider how best to build in a manner that avoided co-ordination between processes and replica computation whilst maintaining a reasonable load balance
  - Considerations of indirect memory accesses to bear in mind

---

- Further work needed on the iterative solver, especially from the geomagnetisism perspective
- Further work on dynamically tuning the prefetching distance
- PETSc OpenMP/MPI hybrid mode

epcc | THE UNIVERSITY OF EDINBURGH