Developing Hybrid OpenMP-MPI Parallelism for Fluidity - Next Generation Geophysical Fluid Modelling Technology

Xiaohu Guo, Gerard Gorman, Andrew Sunderland, Mike Ashworth
ARC, CSE Department, STFC
AMCG, Department of Earth Science and Engineering,
Imperial College London

INTRODUCTION

- An overview of the Fluidity-ICOM Numerical Technology
- Developing hybrid OpenMP/MPI parallelism
  - ✧ Thread Safe Issues and Solutions
  - ✧ Optimization of Memory Bandwidth
- Summary and Conclusions
- Future work

dCSE ICOM Collaborations

- Applied Modeling and Computation Group, Imperial College, London (AMCG, http://amcg.ese.ic.ac.uk/)
- ARC, The Computational Science & Engineering Department (CSED), STFC (http:// www.cse.clrc.ac.uk/)
- National Oceanographic Laboratory, Liverpool (POL, http://www.pol.ac.uk/)
- Edinburgh Parallel Computing Centre, Edinburgh(http:// www.epcc.ed.ac.uk/)

Science & Technology Facilities Council

Motivations for the next generation ocean model

- To resolve a wide range of spatial and temporal scales
- Model internal waves, boundary currents, eddies, overflows, convection events, …, accurately and efficiently within a global and coupled context
- Need for accurate and efficient representation of highly complex domains
- Ability to model interaction of flow with small scale topography, shelf seas, coastal regions, islands, estuaries, harbours,…

An overview of the Fluidity-ICOM Numerical Technologies

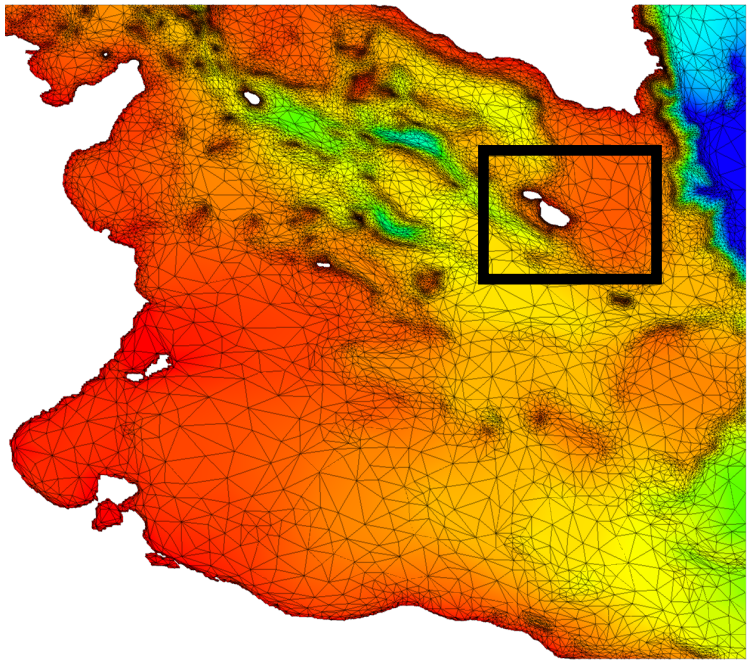
Science & Technology Facilities Council

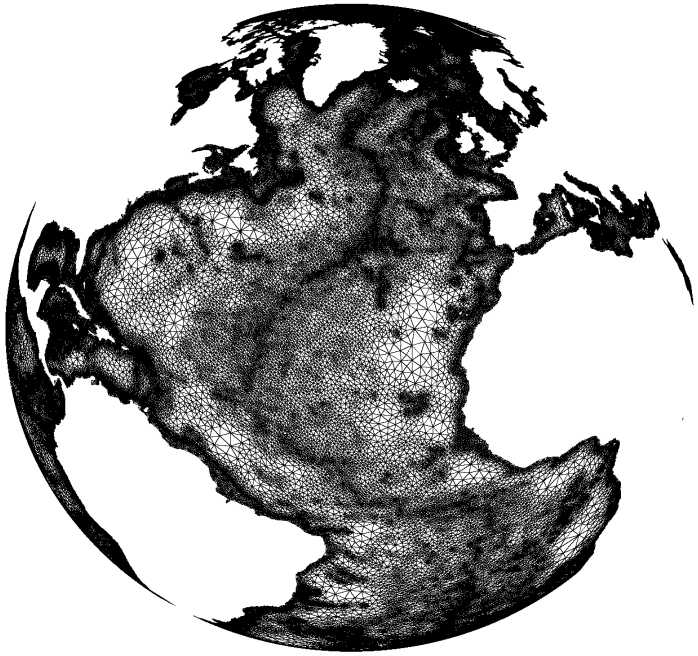
- **Unstructured FEM**

Start with Fluidity – an open source control volume finite element solver for 3D compressible multi-phase fluids. Has been developed by AMCG for more than a decade and is the basis for a range of multi-physics multi-scale applications

Initial mesh generation to follow complex bathymetry and coastlines -- **terrno**

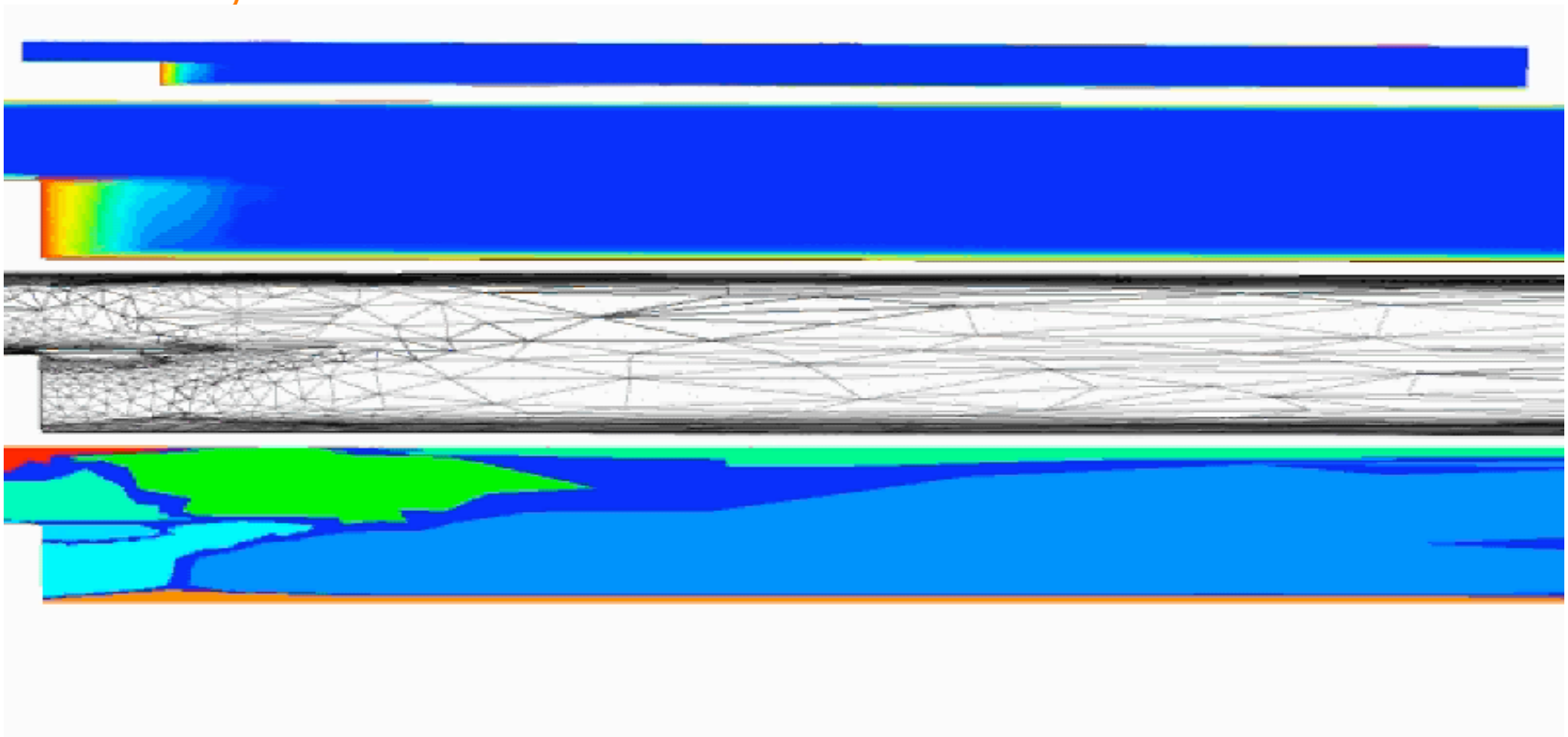
Unstructured meshes are an Ideal choice for representing complex problem domains and a coupled range of scales without the need for grid nesting





Science & Technology Facilities Council

- **Adaptive Mesh, solving from large scales to small scales.**
  - Add an adaptivity library which performs topological operations on the mesh, and mesh movement, to optimise the size and shape of elements in response to error measures
  - Dynamic load balance method -- **Zoltan**



- Fortran, C++, Python, MPI Based
- Open source community model development approach
  - Makes use of open source solutions for I/O, Visualisation, etc
  - Advantage – using latest software features
- The Fluidity source code is hosted on launchpad

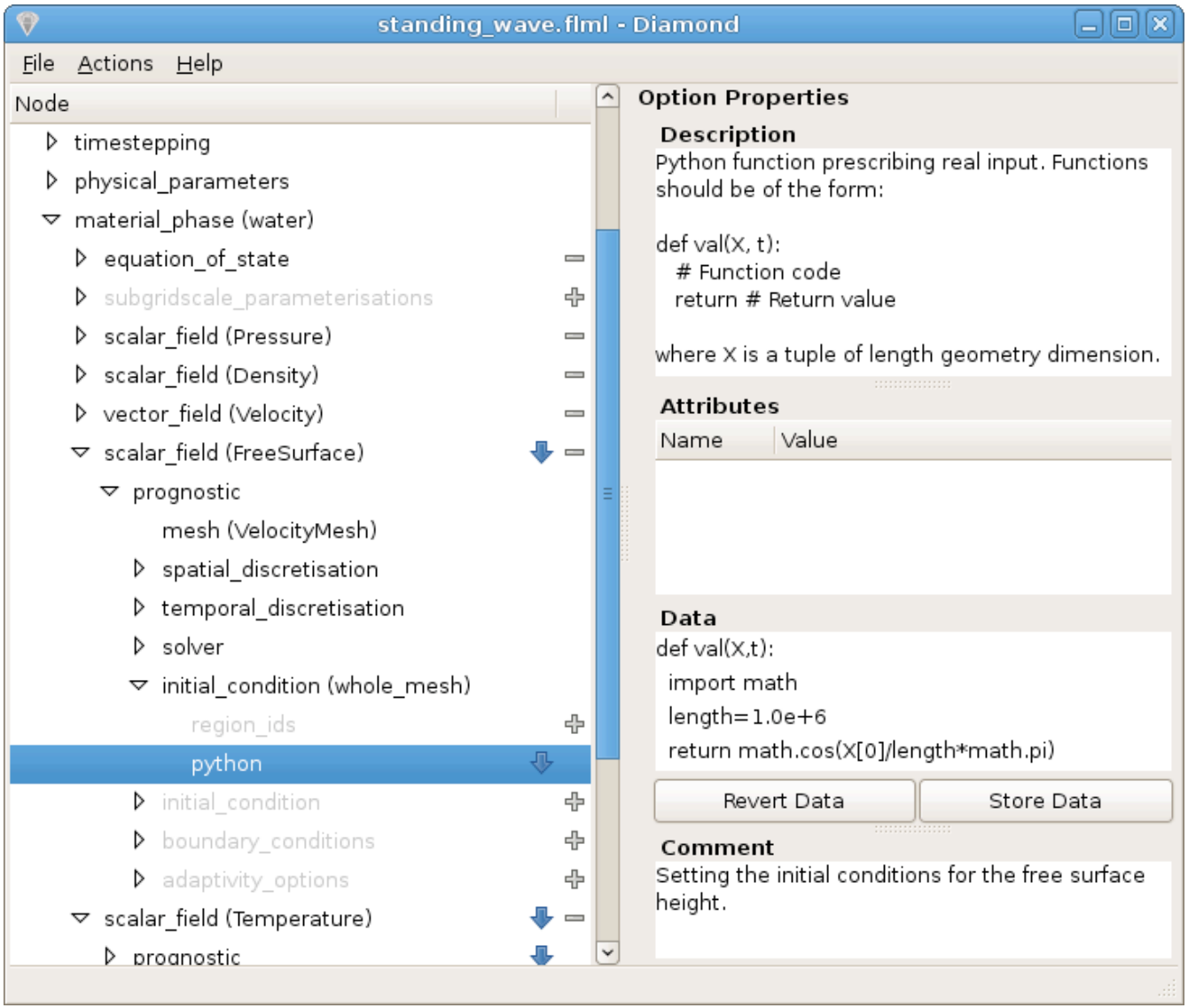
ICOM Software Package Lists

- VTK
- CGNS
- BLAS
- LAPACK
- XML2
- MPI
- PETSc
- Zoltan/ParMetis
- APPACK
- NetCDF
- UDUnits
- Python Development Environments
- Trang
- Spatial-Index
- Fortran 90 Compilers
- C++
- Bazaar

Science & Technology Facilities Council

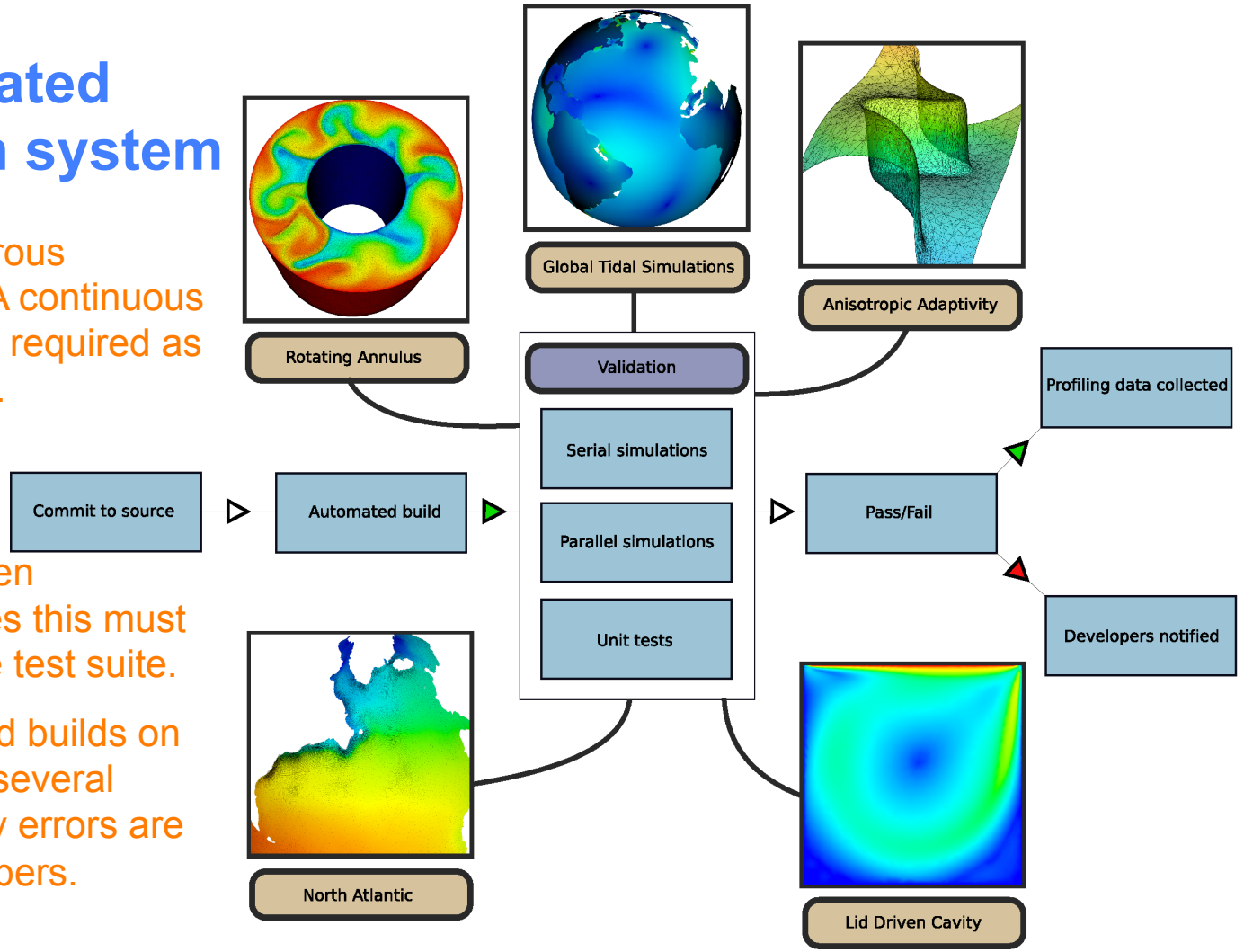
Diamond automatic pre-processing tool

- An xml schema file describes the rules that govern model options
- Diamond uses this to automatically generate a GUI based on the schema
- Options are entered and output as another xml file containing the options values
- This is read into an options library accessible from anywhere in code
- Includes many features, including the ability to define python functions executed at run time



Buildbot: automated code verification system

- All models require rigorous validation/verification. A continuous automated approach is required as the codebase changes.
- A central copy of the source is kept in a bazaar repository. When developers commit changes this must result in a pass of the code test suite.
- Buildbot checks out and builds on various platforms with several different compilers. Any errors are relayed back to developers.

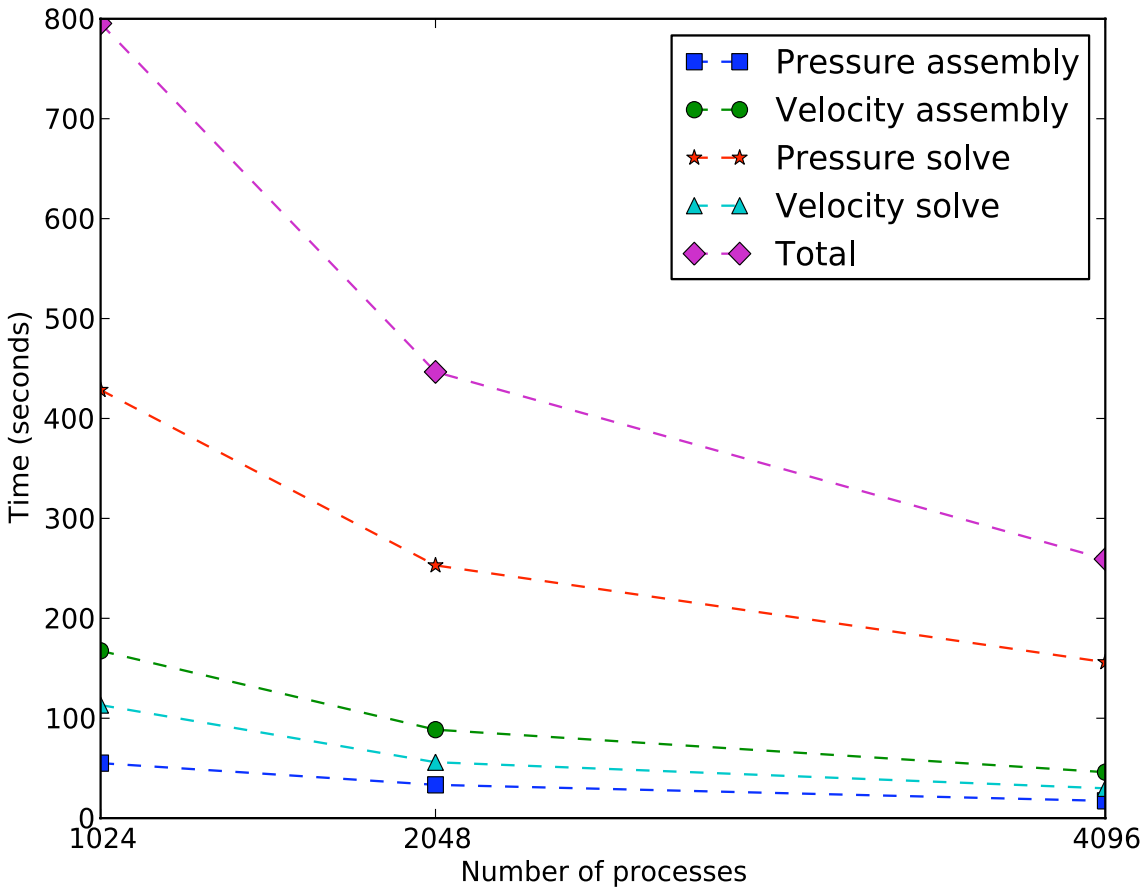


- If a failure is detected in a test problem, the developers are notified details via email.
- Statistical information about code quality is automatically collected from the newly validated code. This allows for the monitoring of performance. Results available via a web interface.
- This modern approach to software engineering has yielded dramatic improvements in code quality and programmer efficiency.

Science & Technology Facilities Council

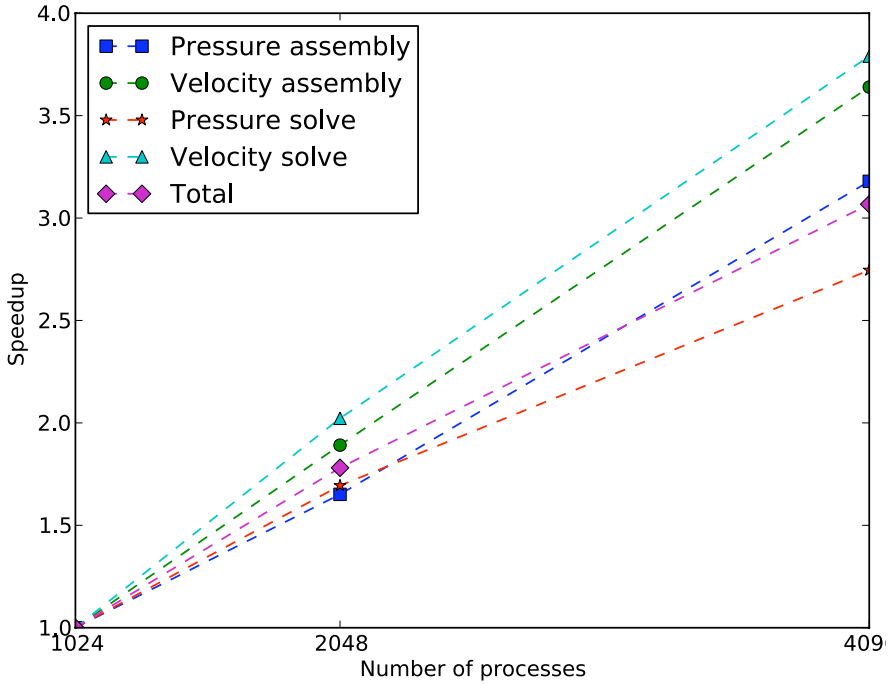
Basic Timings

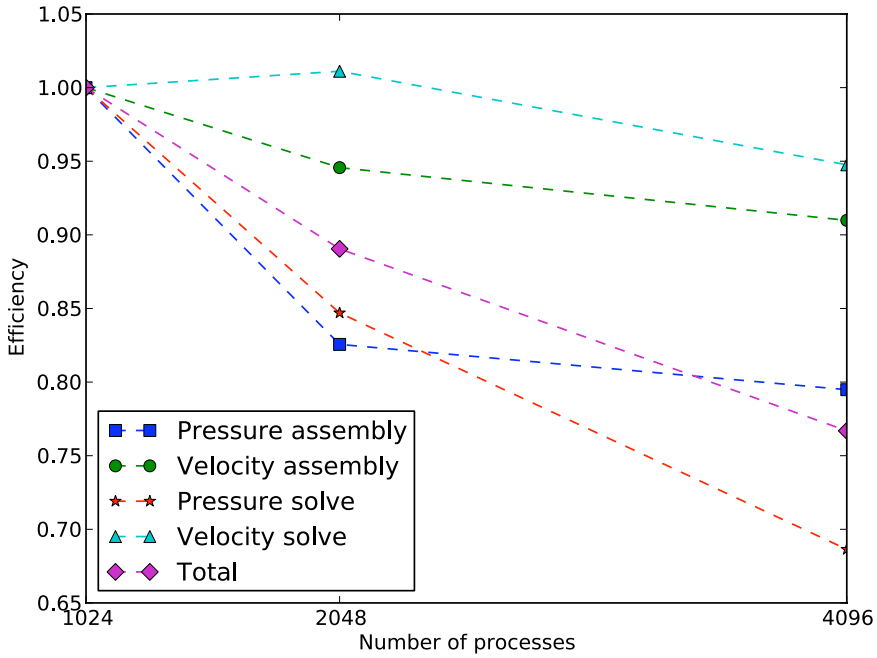
- The solution process consists of the assembly of the linear systems representing the discretised momentum equation and the pressure equation.
- Matrix assembly for pressure and velocity can take more than 30% of the total simulation time with 1024 cores.
- Pressure solver is the main cost
- Matrix assembly phase is expensive
  - Significant loop nesting, where the innermost loop increases in size with increasing quadrature;
  - Indirect addressing (due to unstructured meshes)
  - Cache re-use: renumbering mesh nodes, blocking in matrix assembly



Science & Technology Facilities Council

Speedup and Efficiency





The speedup and efficiency of momentum solver and each of its components

Developing hybrid OpenMP/MPI parallelism for Fluidity/ICOM

Project Rationality

- Further develop Fluidity-ICOM in order to run efficiently on supercomputers comprised of ccNUMA nodes.
- Hybrid OpenMP/MPI decreases the total memory footprint per compute node (the total size of mesh halos increases with number of partitions) and provides memory bandwidth optimization opportunities.
- The use of hybrid OpenMP/MPI will decrease the total volume of data to write to disk, and the total number of metadata operations based on the files-per-process I/O strategy.
- Reduced number of domain partitions benefits many algorithms, e.g. AMG, mesh adaptivity.
- Directive based approach, Same code base, easily port to other platforms, eg: Cray XK6, Intel MIC

Fluidity ICOM Sparse Matrix Assembly

- Using element by element approach
- Sparse matrix storage formats:
  - CSR + diagonal
  - PETSc csr format
- Block assembly
- 30-40% of total computation due to higher order and DG integrations.

Algorithm 1: General Matrix assembly loop

```
Global_matrix ← 0
For e =1, number_of_elements do
      Local_Matrix = Assemble_Element(e)
      Global_Matrix +=  Local_Matrix
enddo
```

An Overview of the OpenMP Implementation

- Working out sparse patterns (element adjacency matrix) for different numerical discretisation method, eg, DG, CG and CV
- Parallelize matrix assembly with colouring method, colouring elements according to their sparse patterns, a loop over colours is added around the main assembly loop.
- The main assembly loop over elements is parallelised using the OpenMP parallel do directive with a static schedule.
- This divides the loop into chunks of size ceiling (number_of_elements/number_of_threads) and assigns a thread to a separate chunk.

Algorithm 2: Threaded Matrix Assembly Loop

```
graph ← create_graph(mesh, discretisation)
colour← calculate_colouring(graph)
k_colours = max(colour)
Global_Matrix ← 0
$!OMP PARALLEL
for k=1, k_colours do
   independent_elements={ e | colour[e] = k }
   for all e in  independent_elements do
          Local_Matrix = Assemble_Element(e)
          Global_Matrix +=  Local_Matrix
    end for
end for
```

Note:

- Generally, the above colouring method tries to colour as many vertices as possible with the first colour, then as many as possible of the uncoloured vertices with the second colour, and so on
- Therefore the number of elements is not balanced between each colour group.
- For OpenMP, it's not a problem as long as each thread has enough work load.
- The performance is not sensitive to the total number of colour groups

Thread Safe Issues and Solutions

Tools for Race Conditions Detecting

- DRD v.s. Helgrind
  - They both have a lot of false positives, also take very long time to generate results for fluidity typical runs(at least ten hours for gyre node performance test case).
  - Need pay a lot of attention to “store” operation, start with “Conflicting Store by…”
  - Very helpful for detecting racing conditions.
- Intel Thread Checker.

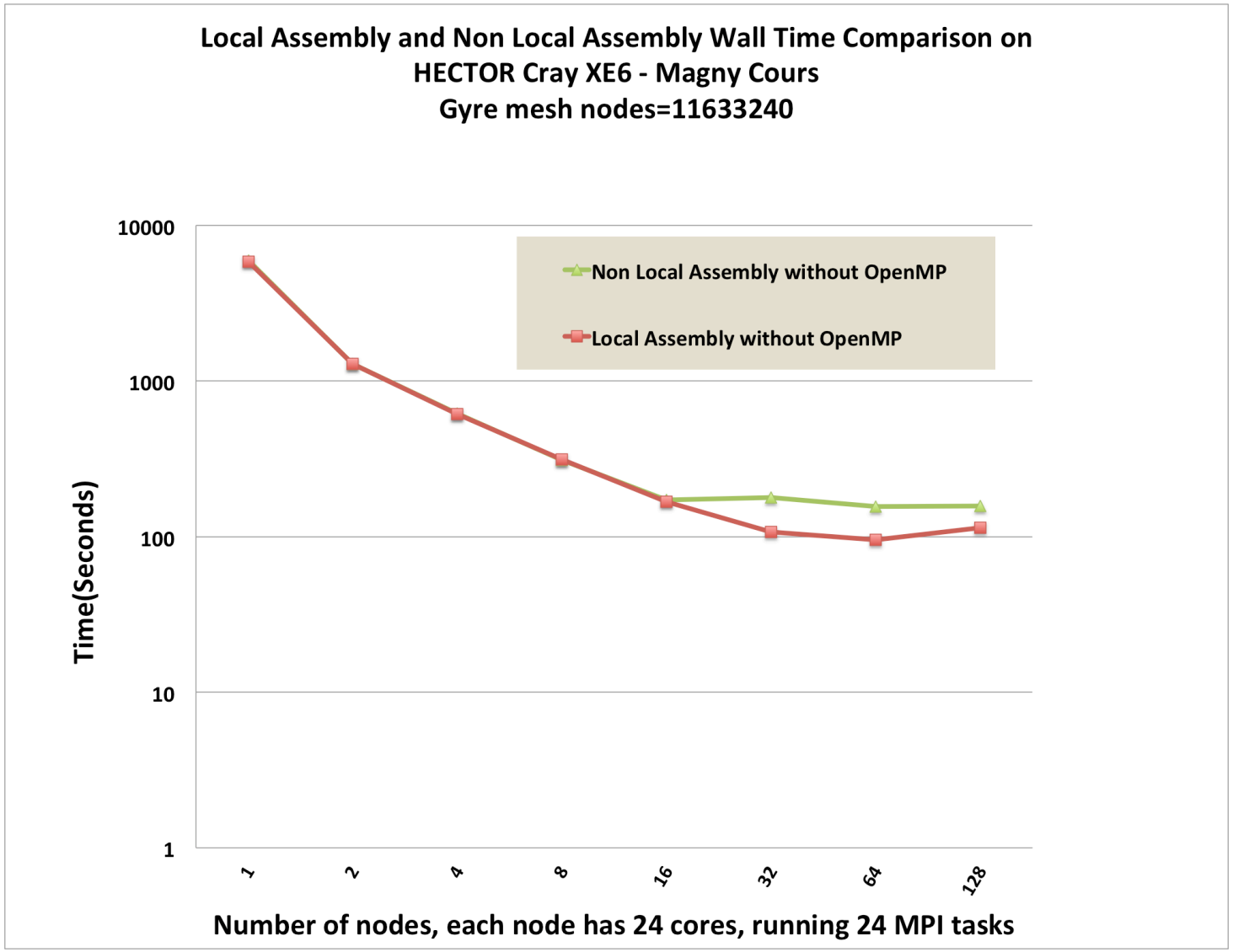
Local assembly v.s. non local assembly

- PETSc Matrix stashing: The stash is used to temporarily store inserted vec values that belong to another processor. During the assembly phase the stashed values are moved to the correct processor -- not thread safe

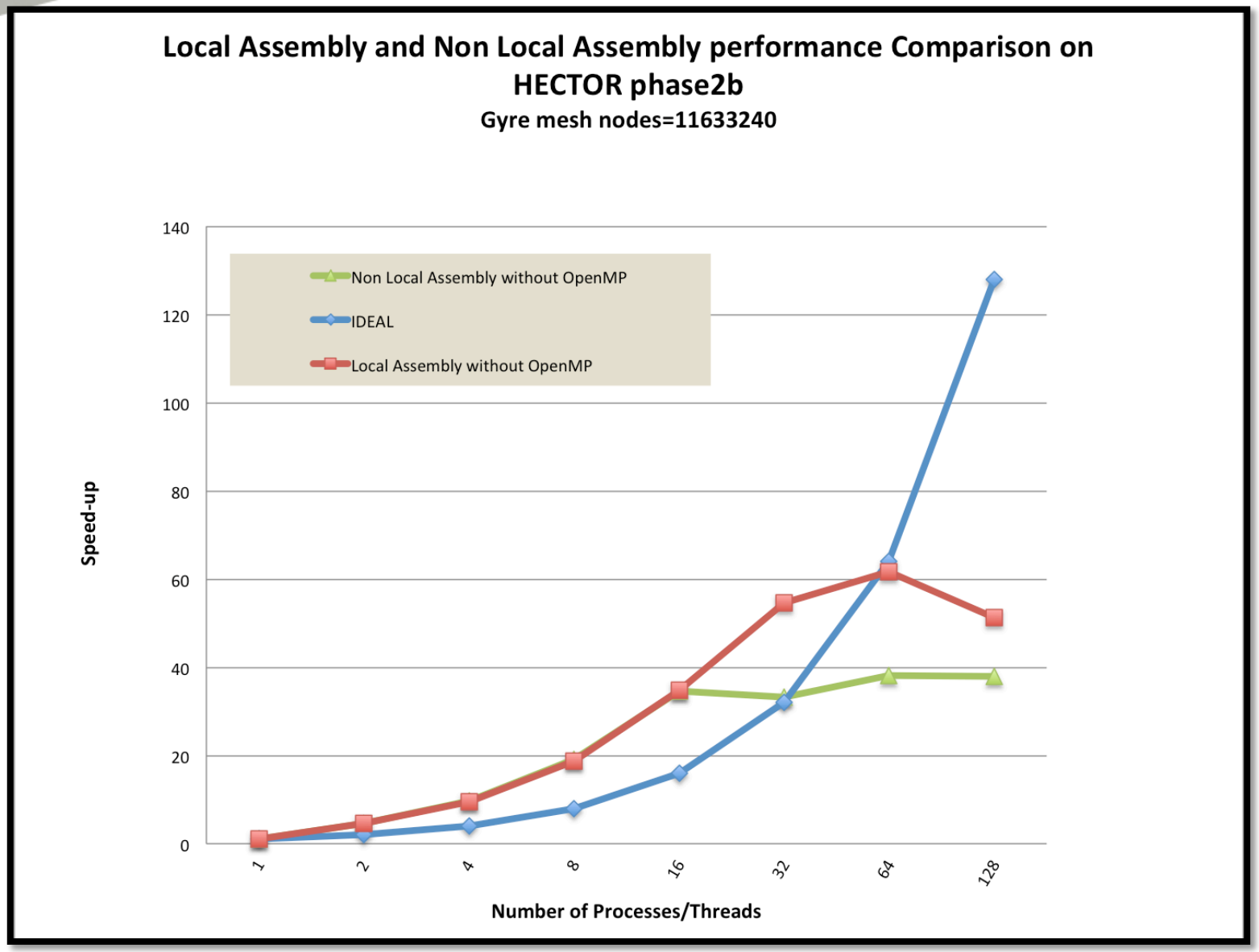
- When **MAT_IGNORE_OFF_PROC_ENTRIES** is set, any **MatSetValues** calls to rows that are off-process will be discarded. This makes matrix assembly much faster as no communications are needed -- recompute rather than communicate

Science & Technology Facilities Council

Local assembly v.s. nonlocal assembly



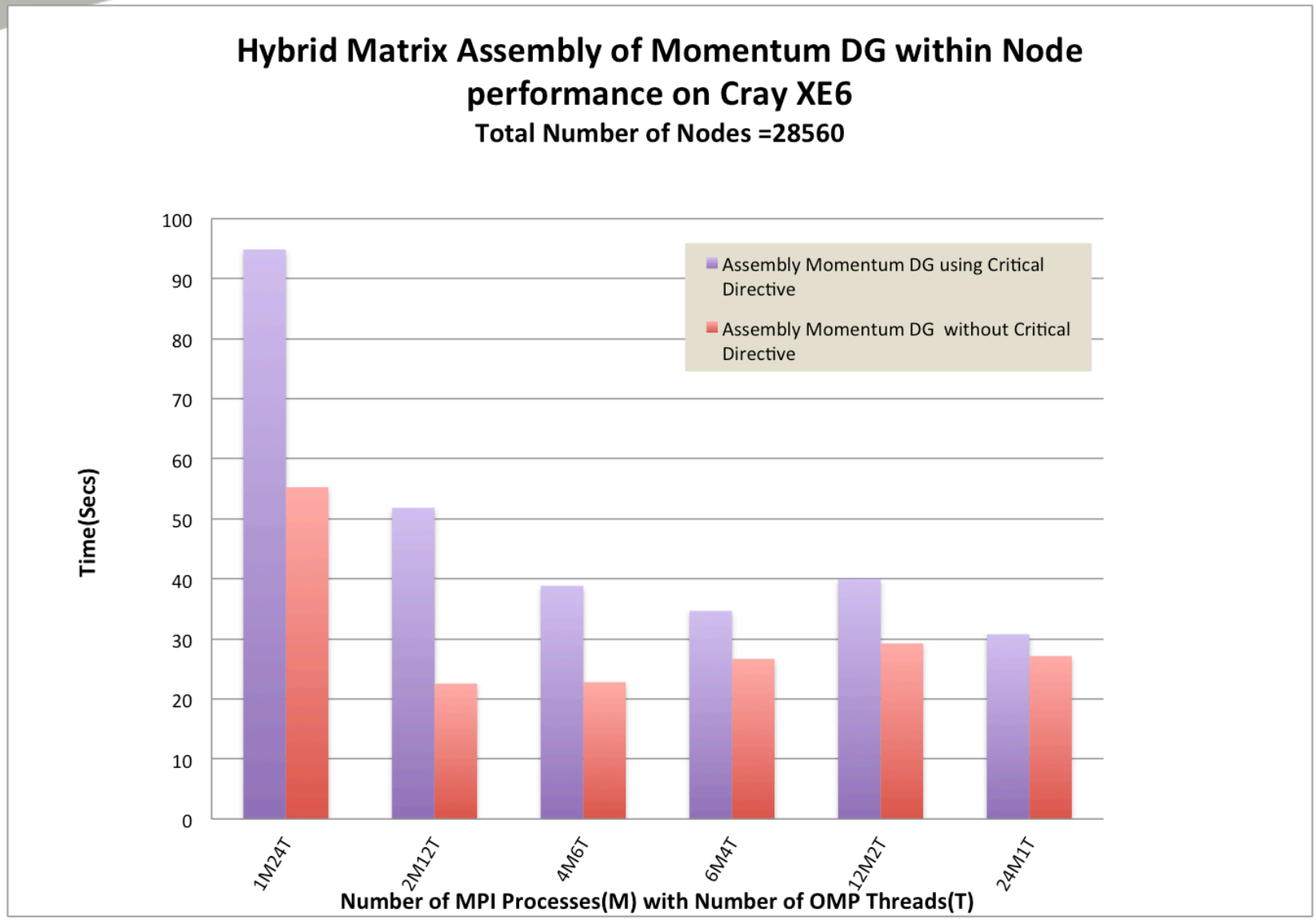
Science & Technology Facilities Council



Thread Safe Issues of Memory Reference Counting

- Any defined type objects in fluidity being allocated or deallocated, the reference count will be plus one or minus one.
- If the objects counter equals zero, the objects should then be deallocated.
- In the element loop, the element-wise physical quantities should not do allocation or deallocation. (But they do, which causes racing conditions for the reference counter.)
- Solutions for the above,
  - add critical directives around reference counter.
  - Move allocation or deallocation outside of element loop

Science & Technology Facilities Council



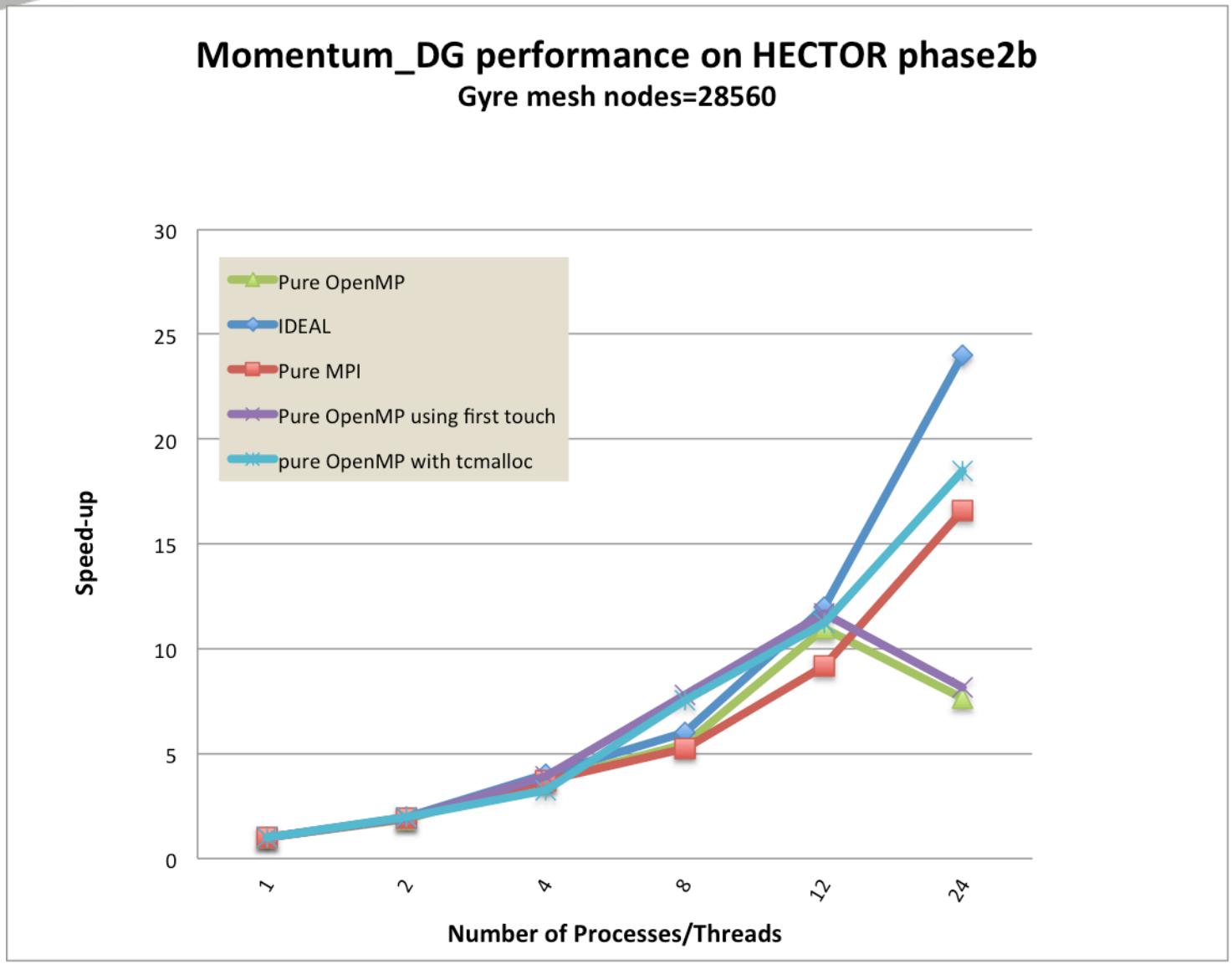
Optimization of Memory Bandwidth

- First touch
  - What this means is that when an application requests memory, the virtual address is initially not mapped to any physical memory.
  - When the application first accesses the memory (read or write), the OS allocates a physical memory region and maps the virtual address to the physical range.
  - The OS typically allocates physical memory from the same NUMA node as the CPU that executed the thread which first accessed the virtual memory block.
- Using NUMA-aware heap memory manager – TCMalloc
  - Reduce the overhead of OS API calls by managing its own memory pool
  - Reduce the lock contention in multi-processor systems
  - Maximize local memory allocations on NUMA systems

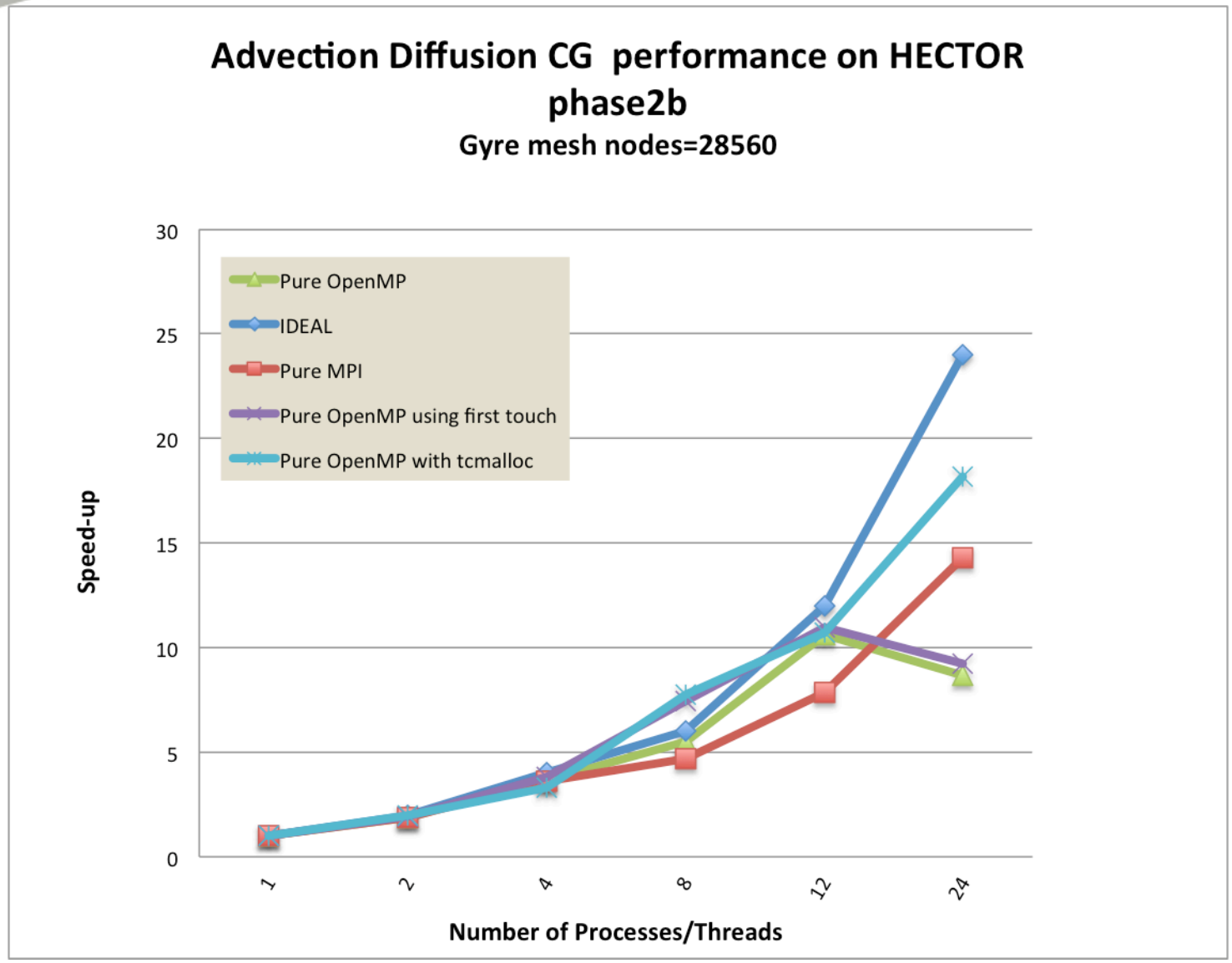
CrayPAT Sample Profiling Statistic of Momentum DG with 24 threads.

```
Samp% | Samp | Imb. | Imb. |Group
      |      | Samp | Samp% | Function
      |      |      |       | PE=HIDE
100.0% | 75471 | --   |  --   |Total
|-------------------------------------------------------------------------
| 95.8% | 72324 | -- | -- |ETC
||------------------------------------------------------------------------
|| 14.6% | 11002 | 0.00 | 0.0% |_int_malloc
|| 13.8% | 10417 | 0.00 | 0.0% |__lll_unlock_wake_private
|| 9.7% | 7284 | 0.00 | 0.0% |free
|| 9.5% | 7172 | 0.00 | 0.0% |__lll_lock_wait_private
|| 6.4% | 4862 | 0.00 | 0.0% |malloc
|| 6.2% | 4674 | 0.00 | 0.0% |__momentum_dg_MOD_construct_momentum_element_dg
|| 4.0% | 3046 | 0.00 | 0.0% |_int_free
|| 3.2% | 2439 | 0.00 | 0.0% |__momentum_dg_MOD_construct_momentum_interface_dg
|| 3.0% | 2272 | 0.00 | 0.0% |_gfortran_matmul_r8
|| 3.0% | 2251 | 0.00 | 0.0% |__sparse_tools_MOD_block_csr_blocks_addto
|| 2.8% | 2090 | 0.00 | 0.0% |malloc_consolidate
|| 2.1% | 1574 | 0.00 | 0.0% |__fetools_MOD_shape_shape
```

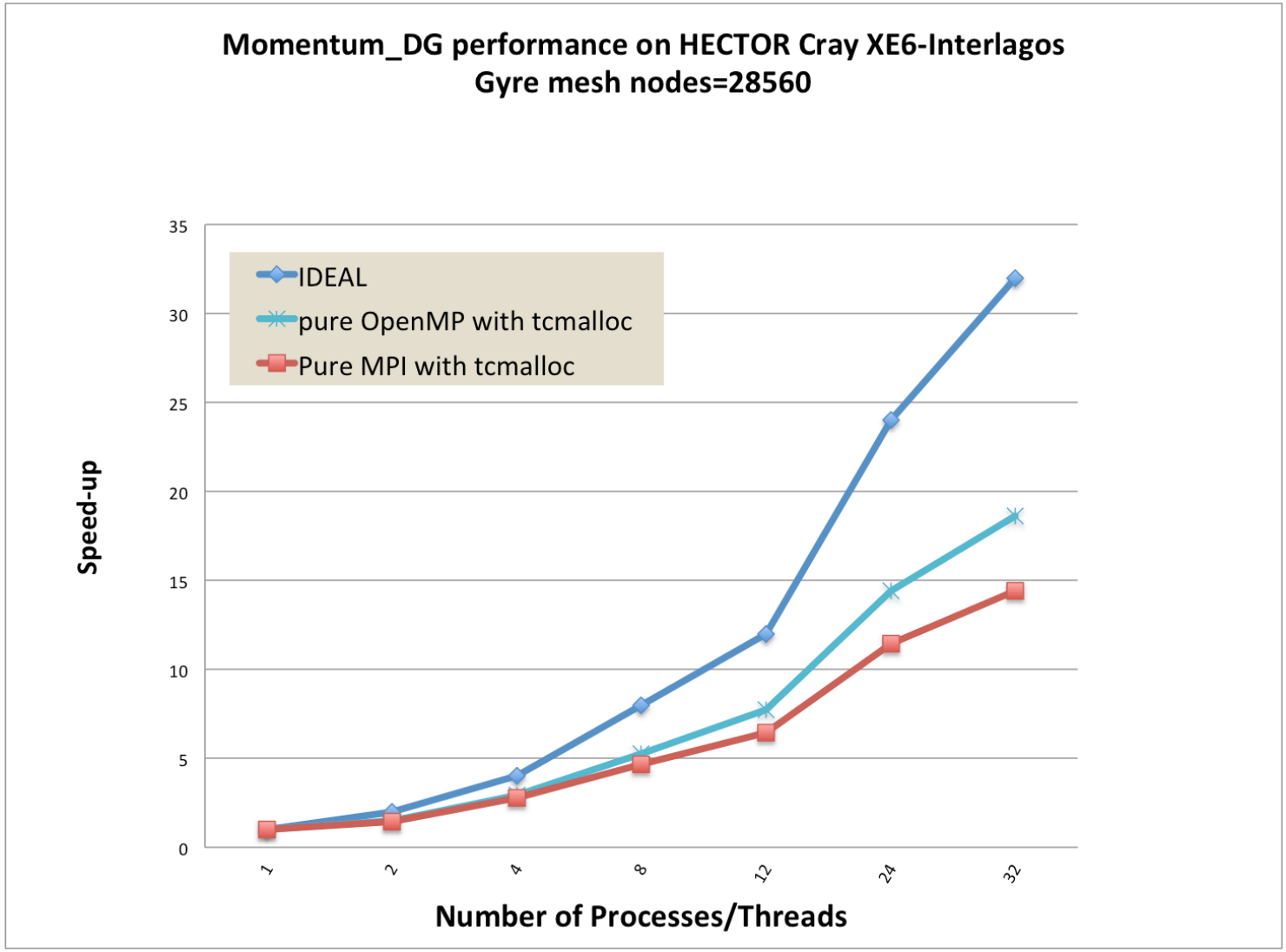
Science & Technology Facilities Council

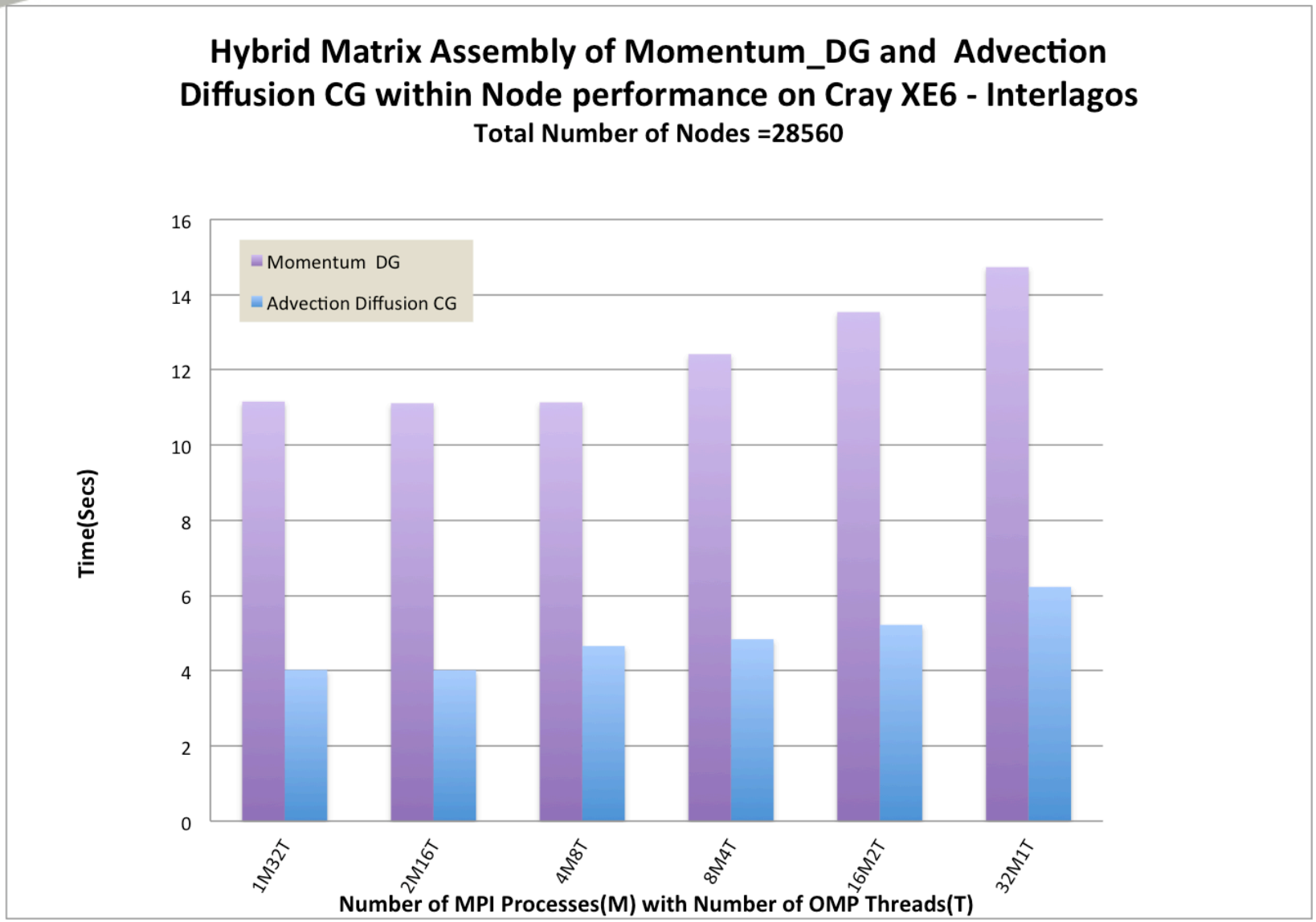


Science & Technology Facilities Council



Science & Technology Facilities Council




Hybrid Matrix Assembly of Momentum_DG and Advection Diffusion CG within Node performance on Cray XE6 - Interlagos
Total Number of Nodes =28560
Momentum DG
Advection Diffusion CG
Time(Secs)
0
2
4
6
8
10
12
14
16
1M32T
2M16T
4M8T
8M4T
16M2T
32M1T
Number of MPI Processes(M) with Number of OMP Threads(T)

Summary and Conclusions

- We have focused on Fluidiy Matrix Assembly.
  - Above performance results indicate that node optimization can be done mostly using OpenMP with an efficient colouring method
  - Regarding Matrix stashing, local assembly performance is better than non local assembly, This makes assembly an inherently local process.
  - Thus focus is on optimizing local (to the compute node) performance, try to avoid use mutual synchronization directives: eg. Critical
  - Improving memory bandwidth usage through NUMA optimisations(eg: first touch, thread pinning) and using NUMA aware heap memory manager can get best performance using pure openmp within the NUMA node

Future Work

- More work on Solvers,
  - investigation of threaded HYPRE that can be called through PETSc.
  - Benchmarking with the PETSc development branch that supports OpenMP.
- Performance Investigation of the fully OpenMP parallelised Fluidity on Intel MIC and Cray XK6
- integrate Fluidity with a newly-developed adaptive mesh library(Pragmatic) which also supports hybrid OpenMP-MPI

Acknowledgements

- The authors would like to acknowledge the support of a **HECToR** *distributed Computational Science and Engineering* award.
- The authors would also like to thank the **HECToR and NAG support team** for their help throughout this work.
- The authors would also like to thank Dr. Lawrence Mitchell and Dr. Michele Weiland for their valuable contributions, Dr. Stephan Kramer for sharing his pearls of wisdom with us during code development and Dr. Stephen Pickles, Dr. Andrew Porter and Dr.Michael Seaton for their valuable suggestions and discussions.

Science & Technology Facilities Council

THANKS for Listening !