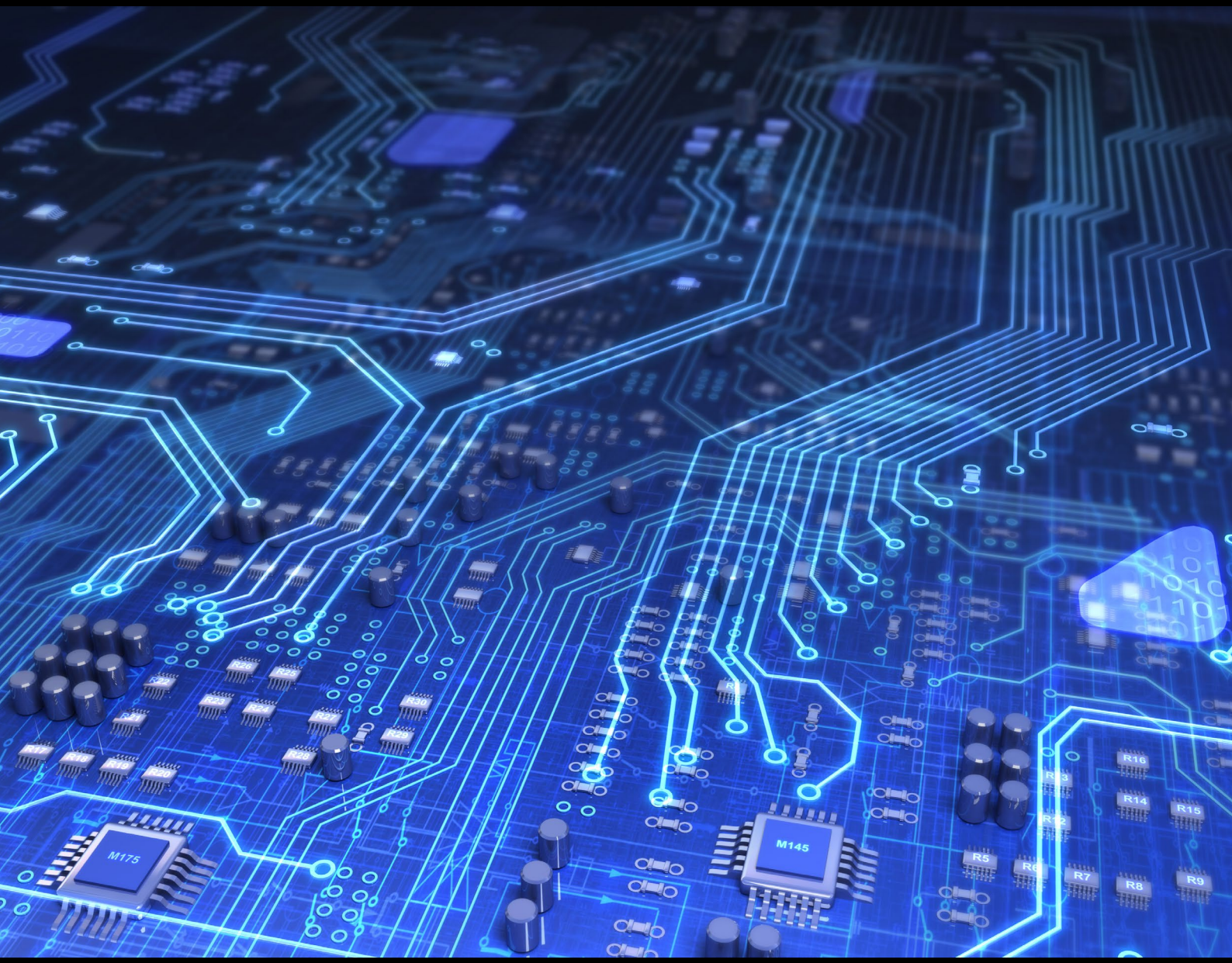
DELL EMC intel

OPENFOAM ON INTEL® XEON PHI™ PROCESSORS



Ravi Ojha
Tata Consultancy Services (ravi.ojha@tcs.com)
Christoforos Hadjigeorgiou
University of Cambridge (UK) (ch741@cam.ac.uk)

UNIVERSITY OF CAMBRIDGE

January 12 2018

Table of Contents

Introduction 3

Project objectives 4

Testing environment 4

Initial pre-processing 4

OpenFOAM optimisation 4

Intel® libhbm 5

Results obtained 5

Conclusions 6

Acknowledgements 7

References 7

1. Introduction

The aim of this report is to provide a basic guide into investigating and optimising OpenFOAM for scaling a realistic model of up to 16 Intel® Xeon Phi™ processor nodes. Although the cases used here are for incompressible flow, the learning outcomes are not exclusive to these input cases or types of simulation.

2. Project objectives

The main objective of the work described here was to investigate the scalability of OpenFOAM on the Intel® Xeon Phi™ processor, using an incompressible flow simulation with an unofficial STL mesh of a Land Rover Evoque mock-up. As the initial model used was not designed for use in a simulation, significant pre-processing was required and optimising the simulation to improve scaling was also investigated. In addition, the benchmark results in this report highlight best practices for running the simulation on Intel® Xeon Phi™ processors.

The DrivAer test case, developed by the Technical University of Munich, was also used as a production-quality benchmark for comparison. These two workloads cover both the RAS and LES models of incompressible solvers that are widely used in the automotive industry. The size of the DrivAer model is 11 million cells whereas the size of the Evoque model was initially 5 million, then increased to 15 million cells.

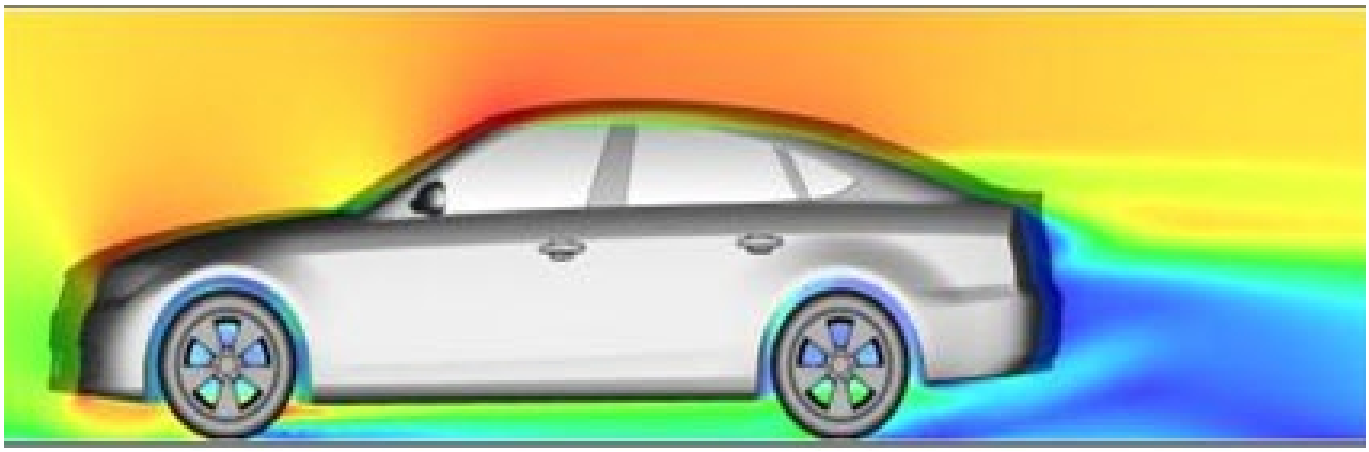


Figure 1: Wind tunnel simulation of DrivAer

3. Testing environment

The processors and clusters used were Intel® Xeon Phi™ 7210 on Peta4-KNL, Intel® Xeon Phi™ 7230 on Dell Zenith and Intel® Xeon Phi™ 7250 on the Intel Endeavour cluster. All the tests performed on Peta4-KNL and Zenith used Intel 2018 Compilers and MPI and the development branch of OpenFOAM.

Intel® C++ Composer and MPI version 2018.0.082 Beta was used on Endeavour. Peta4-KNL nodes were running Scientific Linux 7.3 (kernel 3.10.0-514.32.3.el7.x86 64), Zenith was running Red Hat Enterprise Linux 7.3 (kernel 3.10.0-514.el7.x86 64) and Endeavour Red Hat Enterprise Linux 7.4 (kernel 3.10.0-514.6.2.0.1.el7.x86 64.knl1)

4. Initial pre-processing

The main model used for the work is of a freely available Land Rover Evoque. The initial model contained a large amount of non-manifold edges and intersecting geometry. To repair these issues Blender was used extensively, in conjunction with the snappyHexMesh plug-in, which assists in exporting a suitable snappyHexMesh configuration.

5. OpenFOAM optimisation

HPC community experience tells us that OpenFOAM is known for its weak scaling. With this in mind, and in order to increase scalability to up to 16 nodes, the mesh size of the initial Evoque model was increased from 5 million to 15 million cells. The simplest way to achieve the expansion was to increase the {x,y,z} parameters in the blockMeshDict file. However the mesh size could not be scaled beyond certain limits using these parameters alone, so more fine-grained tweaks to the mesh were performed using snappyHexMesh. The biggest concern when updating the parameters in the snappyHexMesh dictionary is in maintaining the integrity of the mesh so that the skewness of the entire mesh does not grow beyond

acceptable limits. A generated face is skewed if it differs from an ideal face beyond a certain threshold, which was tested using the checkMesh tool that evaluates the mesh and reports issues related to it. Table 1 provides an overview of the changes, which are separated into two types: mesh size refinement and mesh quality. The scaling from 5 million to 15 million cells was an incremental process, validated with checkMesh at each step.

The prime objective of increasing the mesh size was to understand how well OpenFOAM scales on Intel® Xeon Phi™ processor, which can also be performed if a more fine-grained solution is required.

Using featureAngle to lower the intersecting angle of cells at which refinement is performed improved the quality of the mesh. The resolveFeatureAngle is used to set the angle at which a local curvature threshold is identified - a lower value means that a higher number of smaller block cells is used, enabling a better description of the geometry of a curve. Also, free standing zone faces - faces that share the same owner and neighbour cell zone: i.e. a cell which is attached to a larger cell zone but which also has a face between the cell and the cell zone - will be snapped to the surface of the mesh, potentially reducing the quality. implicitFeatureSnap and multiRegionFeatureSnap detect geometric features and features between multiple surfaces, which are used to snap or adapt the created mesh to the input geometry. To increase the mesh size, we decreased the minimum threshold for cell refinement (minRefinementCells), with cell refinement being performed until the minimum value is reached.

Parameter	Default	Optimal	Comment
minRefinementCells	10	5	Lower value → finer mesh
maxLoad Unbalance	0.10	0.05	Lower value → finer mesh
nCellsBetweenLevels	3	5	Increase mesh size
resolveFeatureAngle	30	1	Lower angle → reduce skewness
allowFreeStandingZoneFaces	True	False	Remove scope of disconnected faces
implicitFeatureSnap	False	True	Improve mesh quality
multiRegionFeatureSnap	False	True	Improve mesh quality
nSurfaceLayers	1	4	Higher value → increase mesh size
FeatureAngle	60	1	Increase mesh quality
nSmoothSurfaceNormals	1	5	Increased smoothing → increased size
nSmoothNormals	3	5	Increased smoothing → increased size
maxThicknessToMedialRatio	0.3	1	Improve mesh quality

5.1 Intel® libhbm

When running with the MCDRAM (Colfax Research [2016]) in flat mode, libhbm, which can be obtained from the OpenFOAM-dev-Intel branch on Github, was used. libhbm (Intel® [2017]) provides a significant performance boost and is used to create a fixed-size memory heap for a process and to set a threshold for deciding whether to use the heap or the main DDR memory. Since MCDRAM offers a bandwidth of 400 GB/s it makes sense to allocate larger data structures to it, while smaller data structures could be allocated to DDR, which has better latency than MCDRAM.

5.2 Results obtained

Table 2: Evoque 5M cells - Intel® Xeon Phi™ Processor 7250

Nodes	Flat 68	Cache 68
1	203	214
2	135	200

Table 3: Evoque 15M cells, Intel® Xeon Phi™ Processor 7250

Nodes	Flat 68	Flat 136
1	1073	1667
2	474	443
4	264	377
8	193	382
16	131	536

The result shown in the tables are the time in seconds - i.e. configuration 68 and 136 processes per nodes

Table 4: Evoque 15M cells, Intel® Xeon Phi™ Processor 7230

	Flat		Cache	
Nodes	64	128	64	128
1	1402.65	NA	1215	1017.34
2	621.73	NA	639.92	544
4	334.99	455.75	342.07	448.96
8	211.68	419.25	214.61	411.3
16	178.07	547.08	176.62	528.22

Table 5: Evoque 15M cells, Intel® Xeon Phi™ Processor 7210

	Flat		Cache	
Nodes	64	128	64	128
1	1506.16	1120.43	1343.36	1119.46
2	649.27	634.88	677.34	656.73
4	407.47	458.45	422.86	468.43
8	231.75	513.93	238.92	527.16
16	183.19	517.89	188.34	564.49

Table 6: DrivAer, Intel® Xeon Phi™ Processor 7250

	Flat	
Nodes	68	136
1	542	683
2	286	310
4	173	124
8	124	535
16	111	1039

Table 7: DrivAer, Intel® Xeon Phi™ Processor 7230

	Flat		Cache	
Nodes	64	128	64	128
1	NA	NA	642.01	584.11
2	345.12	363.15	361.32	414.66
4	205.79	364.11	219.93	381.96
8	151.3	623.75	160.04	598.09
16	133.27	1071.01	134.06	1077.36

Table 8: DrivAer, Intel® Xeon Phi™ Processor 7210

	Flat		Cache	
Nodes	64	128	64	128
1	751.18	696.24	707.36	600.18
2	369.35	401.96	401.24	413.79
4	224.73	390.78	230.58	397.11
8	168.86	622.45	171.38	653.46
16	146.76	1055.16	147.71	1065.71

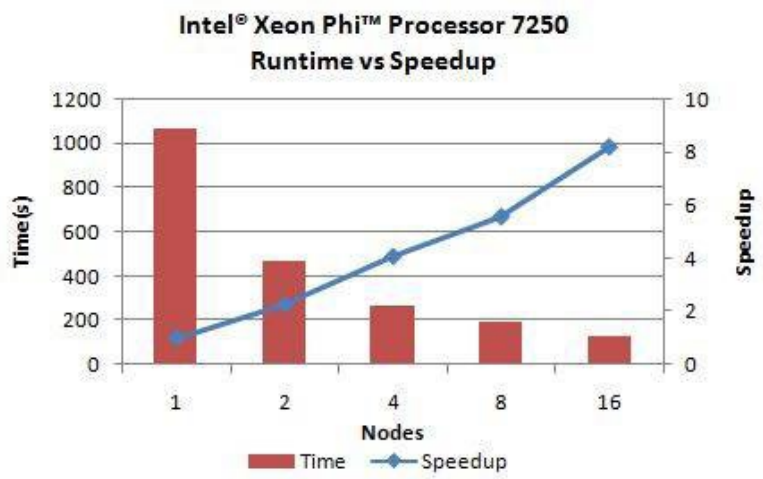


Figure 2: Optimal Evoque time and speed-up

6. Conclusions

For input cases that exceed the 16 GByte capacity provided by the MCDRAM module, cache mode brings performance benefits for single node (see Table 4). The Evoque 5M input case has a memory footprint of less than 16 GBytes and fits completely inside the MCDRAM memory. In such cases the flat mode outperforms cache mode (see Table 3).

On single node, using two ranks per core of Intel® Xeon Phi™ usually gives better performance. For example, DrivAer with 11M cells (table 8) requires about 600.18 seconds using 128 MPI processes on a node compared to 707.36 seconds using 64 MPI processes. In the case of multi-node simulations, the communication overhead (8+ nodes) exceeds the computation cost so some reduction in scaling is observed.

Conclusions have been verified using Intel® Trace Analyser and Collector, which show that for 8 nodes, communication consumes 52% of the runtime in the DrivAer input case and a third of the runtime in the Evoque input case.

The scaling tests of the Evoque case, both on 5M and 15M cell models, prove that OpenFOAM exposes a good weak scaling behavior, as long as sufficient computation is allocated per core of Intel® Xeon Phi™ processor. It is worth noting that in automotive workloads, the mesh size usually ranges from a few hundred million cells to a billion cells and the results presented above also show a super-linear scalability on two and four nodes of the Intel® Xeon Phi™ processor when compared to single node. So for large simulations the Intel® Xeon Phi™ processor could be a good candidate for scaling-up the relevant industry-scale workloads and reducing simulation time-to-solution.

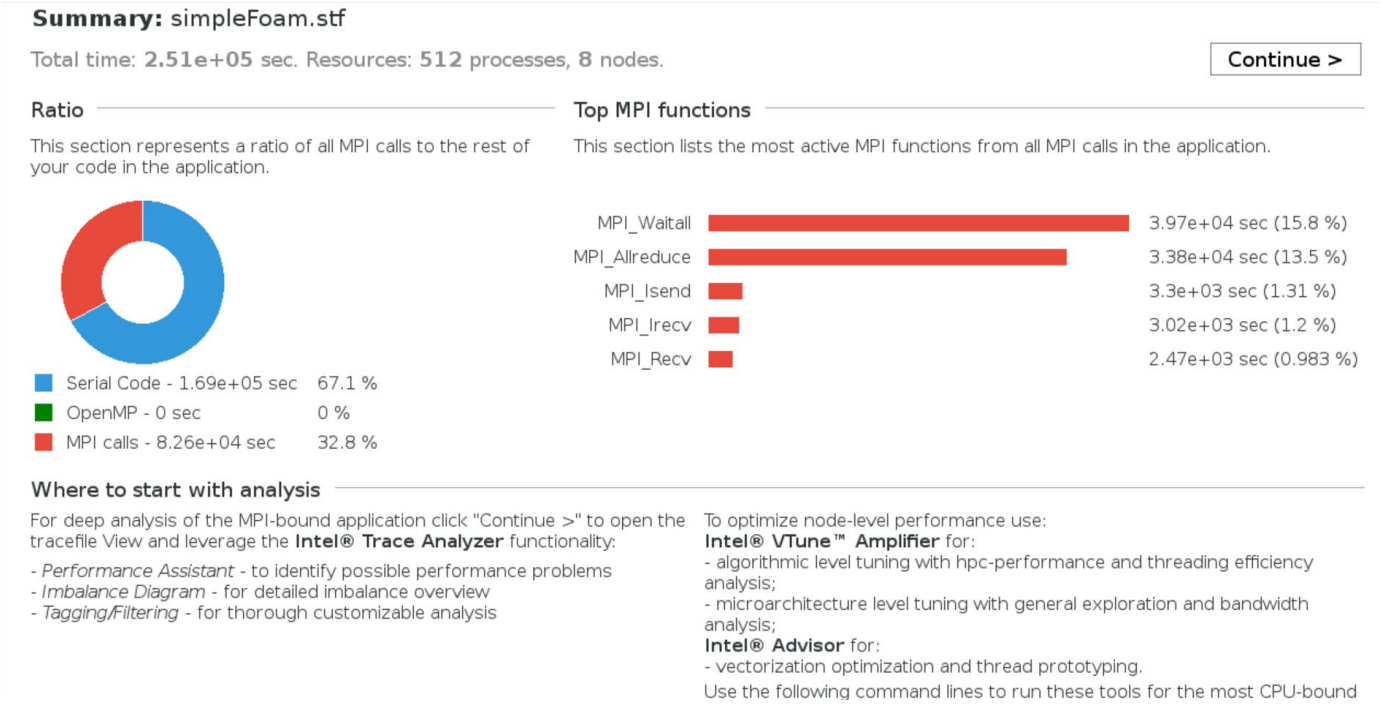


Figure 3: Intel® Trace Analyser and Collector

7. Acknowledgements

We would like to acknowledge Gopal Shinde from Tata Consultancy Services for his valuable contribution towards the mesh refinement process. We also acknowledge the Dell HPC Engineering group in Austin (USA) for providing support and access to Dell internal compute facilities based on Intel® Xeon scalable processors and Intel® Xeon Phi™ processors.

References

Intel® OpenFOAM repository - Intel® specific developments, 2017.

Colfax Research. MCDRAM as high-bandwidth memory (hbm) in Knights Landing processors: Developer's guide, 2016.