Evaluating the Maturity of OpenFOAM Simulations on GPGPU for Bio-fluid Applications

Ahmet Duran
Istanbul Technical University, Department of Mathematics
http://web.itu.edu.tr/aduran

Senol Piskin
Koc University, Department of Mechanical Engineering

Mehmet Tuncel
Istanbul Technical University, Department of Mathematics

EMiT Conference 2016

To cite: Ahmet Duran, Senol Piskin, and Mehmet Tuncel, Evaluating the maturity of OpenFOAM simulations on GPGPU for bio-fluid applications, Proceedings of the Emerging Technology (EMiT) Conference, pp. 11-14, Barcelona Supercomputing Center, Spain, 2-3 June 2016.

Outline

- We deal with the computational challenges for bio-medical fluid flow simulations and an *OpenFOAM 2.2.2* solver, *icoFoam*, for the large matrices coming from the simulation of blood flow in arteries on different HPC clusters
- The flow problem produced various matrices as the time advances in simulation.
- We examined the behaviour of the solvers for ill-conditioned matrices
- We compared the CPU performance of the iterative solver *icoFoam* and the hybrid parallel codes (MPI+OpenMP) of a direct solver *SuperLU_DIST 4.0* (Li at al. 1999, updated 2014) at TGCC Curie (a Tier-0 system) thin nodes at CEA, France
- We compared the performance of the hybrid parallel codes of MPI+OpenMP+CUDA versus MPI+OpenMP implementation of SuperLU_DIST 4.0 at TGCC Curie (a Tier-0 system) hybrid nodes of CPU + GPU at CEA, France
- We discuss the performance, scalability and robustness of OpenFOAM on GPGPU cluster
- We present our results regarding the speed-up of the solvers for the large matrices of size up to 20 million x 20 million

Challenges

- The benefits versus drawbacks of hybrid nodes
- There are tradeoffs using GPU accelerators especially for the software packages or applications where it is not possible to fit the whole part into GPU
- While it is expected to obtain a reduced time due to the accelerator, there would be communication over-head between the various processors and the GPU accelerators, as well
- Therefore, it is important to obtain a feasible/optimal proportion of the tasks to MPI, OpenMP, and CUDA/OpenCL usages in emerging CPU+GPU systems
- For example, it is not possible to do everything only in GPU for a complex algorithm like SuperLU_DIST
- Therefore hybrid nodes like Curie hybrid nodes at CEA in France provide opportunity

Configuration of Curie

- The Curie supercomputer offers three different kind of compute nodes: thin nodes, super fat nodes and hybrid nodes.
- The compute nodes are connected through a QDR InfiniBand network.
- This high throughput and low latency network is used for I/O and communications among nodes of the supercomputer.
- The topology of this InfiniBand network is a full fat tree.
- http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm

- **Thin nodes** for regular computation
  - Partition name: standard
  - CPUs: 2x 8-cores SandyBridge@2.7GHz (AVX)
  - Cores/Node: 16
  - Nodes: 5040
  - Total cores: 80640
  - RAM/Node: 64GB
  - RAM/Core: 4GB
- **Hybrid nodes** for GPU computing and graphical usage
  - Partition name: hybrid
  - CPUs: 2x 4-cores Westmere-EP@2.67GHz + 2x GPU Nvidia M2090
  - Cores/Node: 8
  - Nodes: 144
  - Total cores: 1152 (+ 288 GPU)
  - RAM/Node: 24GB (+6GB GPU)
  - RAM/Core: 3GB
- See http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm

Energy requirements for thin nodes versus hybrid nodes

- A diversification of hardware solutions based on the application capability may be needed in order to attain a good efficiency (see N. Meyer et al. 2013 and J. David et al. 2013)
- While the compute partition of Curie thin nodes having total of 80,640 cores consumes 2132 kW, the partition of Curie hybrid nodes having total of 288 Intel® + 288 Nvidia processors uses 108.80 kW as the total power (see TOP500 Supercomputing sites [8] and the Green500 List [9])
- The partition of Curie hybrid nodes outperforms the Curie thin nodes when the energy efficiency is compared in terms of performance per watt and the rates of computation are 1,010.11 MFLOPS/W and 637.43 MFLOPS/W, respectively

Main tasks of the simulation

- We achieved scaled speed-up for large matrices up to 64 million x 64 million matrices and speed-up up to 16384 cores on Curie thin nodes (see Duran et al, J. of Supercomputing, 2015).
- We generated a structured mesh by using blockMesh as a mesh generator tool.
- To decompose the generated mesh, we employed the decomposePar tool.
- After the decomposition, we used icoFoam as a flow simulator/solver tool.
- we examined *OpenFOAM 2.2.2* "*icoFoam*" simulator with an iterative solver such as diagonal incomplete LU preconditioned bi-conjugate gradient in addition to direct solvers such as distributed *SuperLU 4.0* (see [2]).
- The flow problem produced various matrices as the time advances in simulation.
- The solution of the matrices obtained after each time step can be more challenging due to the changing structure of the matrices.
- This change may be caused by mess change or flow variable change.
- The solution time of the matrices can increases as the time advances in simulation

Flowchart of the approach

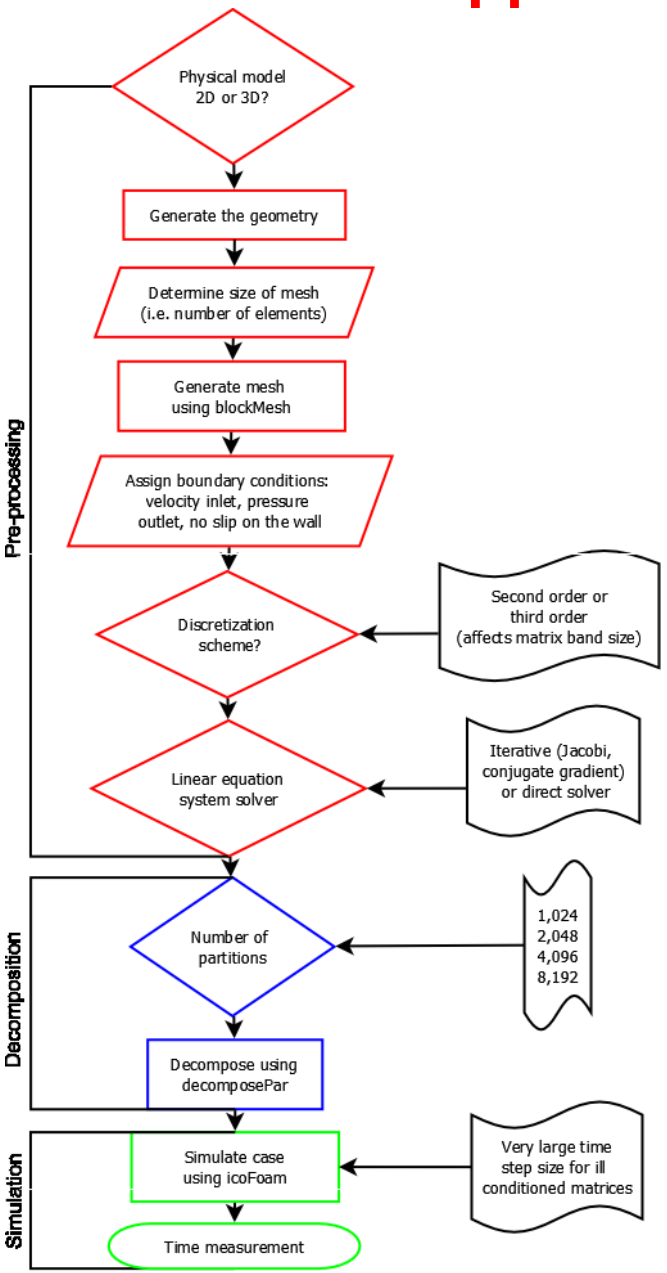


Table I. Description of matrices

	N	NNZ	NNZ/N	Origin
mC_8M	8,000,000	39,988,000	4.999	ITU Mathematics
mC_16M	16,000,000	79,984,000	4.999	ITU Mathematics
mC_6M_D	6,000,000	41,800,000	6.967	ITU Mathematics
mC_8M_D	8,000,000	55,760,000	6.970	ITU Mathematics
mC_8M_n	8,000,000	39,988,000	4.999	ITU Mathematics
mC_16M_n	16,000,000	79,984,000	4.999	ITU Mathematics
mC_20M_n	20,000,000	99,982,000	4.999	ITU Mathematics
mC_6M_n_D	6,000,000	41,780,000	6.963	ITU Mathematics
mC_8M_n_D	8,000,000	55,760,000	6.970	ITU Mathematics
mC_10M_n_D	10,000,000	69,660,000	6.966	ITU Mathematics

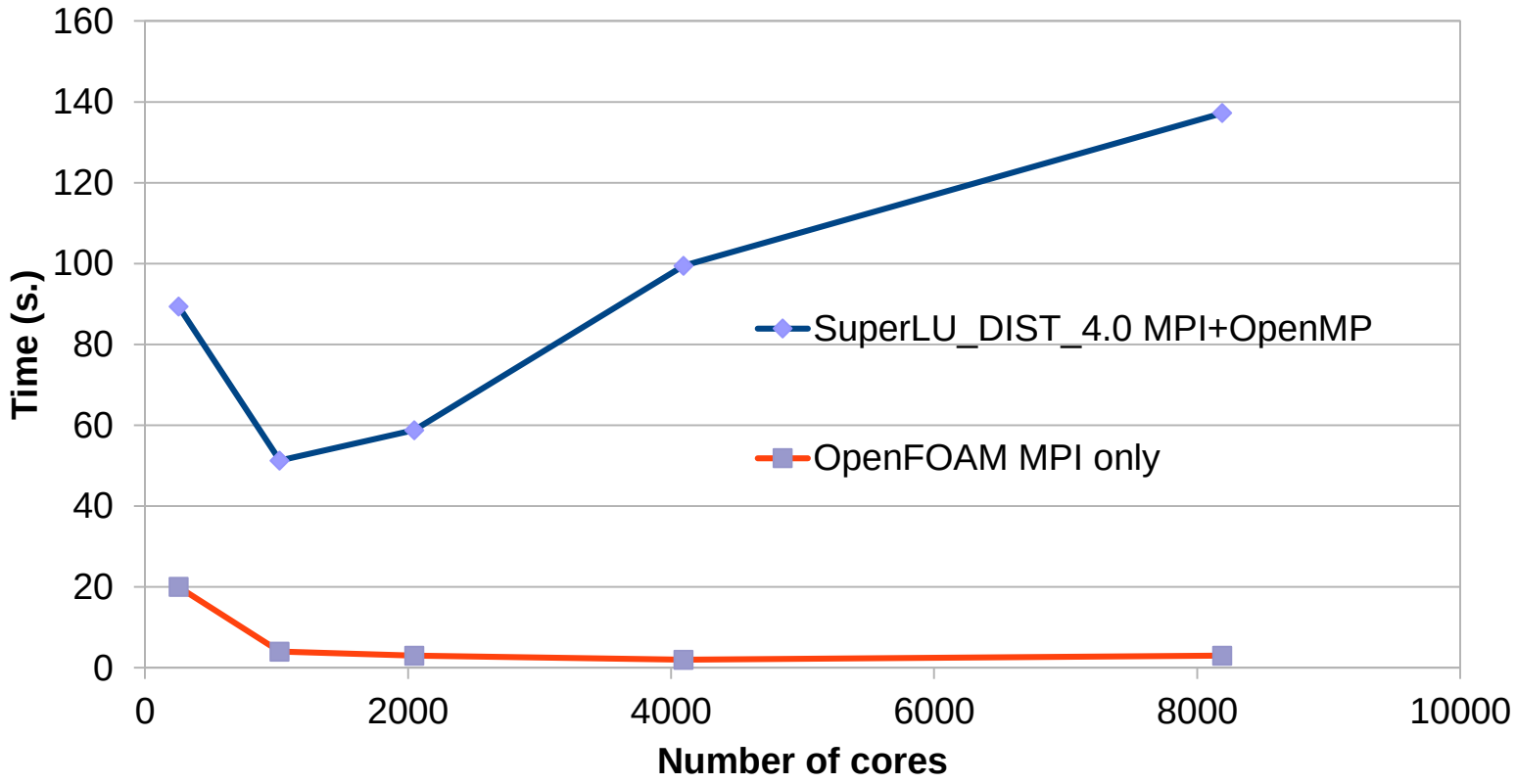
The matrices via simulation

- Here, the solver refers to not only linear system solver but also Navier Stokes solver and simulator.
- **The first four matrices** in Table 1 are obtained at time 0.00005 (s) of the simulation where the time step size is 0.00005 (s), as in [1] (see Duran et al, J. of Supercomputing, 2015).
- Unlike [1], **the last six matrices** in Table 1 are encountered at the third time step, at time 0.012 (s) of the simulation where the time step size is 0.004 (s).
  - This is a relatively large time step size for such a very small mesh size.
- Thus, we obtained challenging ill-conditioned matrices. Almost 5 or 7 banded sparse matrix occurs at each time step.

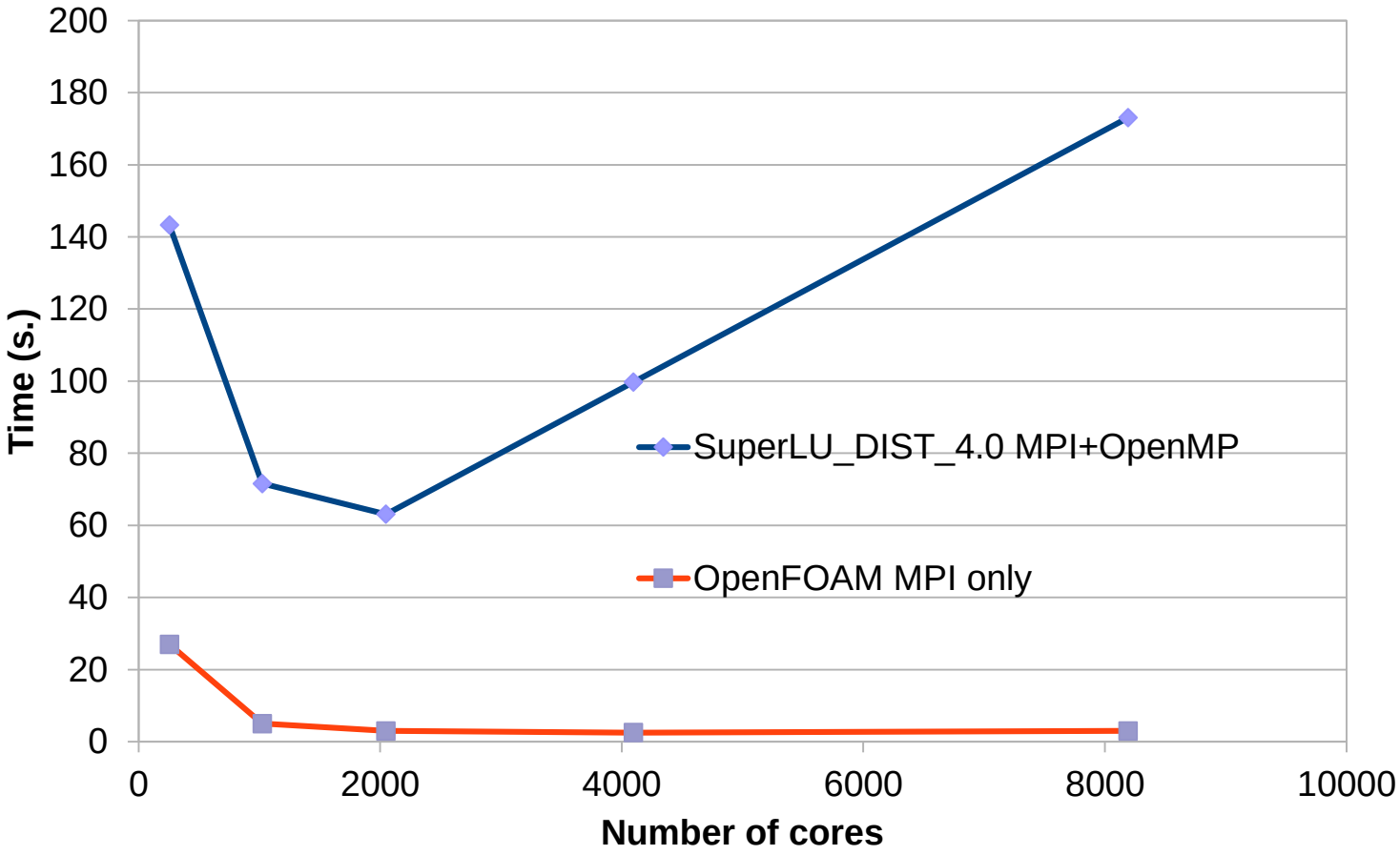
Thin node results

- We compared the CPU performance of the iterative solver *icoFoam* and the hybrid parallel codes (MPI+OpenMP) of a direct solver SuperLU_DIST 4.0 at TGCC Curie (a Tier-0 system) thin nodes at CEA, France
- The following figures show the wall-clock time comparisons of the solvers, excluding the refinement time, for mC_16M_n and mC_20M_n on Curie thin nodes, respectively
- The iterative solver with a diagonal incomplete LU preconditioned bi-conjugate gradient outperforms the direct solver *SuperLU_DIST 4.0* for the simulation matrices

Wall-clock time comparison of the solvers for mC_16M_n on Curie thin nodes



Wall-clock time comparison of the solvers for mC_20M_n on Curie thin nodes



Hybrid node results using MPI+OpenMP+CUDA

- We compared the performance of the hybrid parallel codes of MPI+OpenMP+CUDA versus MPI+OpenMP implementation of SuperLU_DIST 4.0 at TGCC Curie (a Tier-0 system) hybrid nodes of CPU + GPU at CEA, France.
- Table 3 shows the performance results for the ten simulation matrices.
- For example, we observe a linear speed-up of the direct solver up to 512 cores for both implementations for mC_20M_n on Curie hybrid nodes.
- Generally, we see that MPI+OpenMP implementation outperforms the hybrid of MPI+OpenMP+CUDA for this set of simulation matrices due to several overheads coming from CUDA implementation for the direct solver algorithm.
- It is not possible to put everything only in GPU for *SuperLU_DIST*. Therefore, the tasks should be proportioned to MPI, OpenMP, and CUDA/OpenCL.
- In *SuperLU_DIST 4.0*, cuBLAS library execution is one of the most time consuming tasks performed in GPU in order to gain from explicit parallelization.
- On the other hand, there are overheads such as data transfer on PCIe between host and device memory (CPU and GPU) and new data structure changes related to data packing and scattering.

Hybrid node results - continued

- *SuperLU* is a complex algorithm and it is challenging to select the right combination for better intra-node communications and inter-node communications within CPU+GPU heterogeneous systems, under current technology limitations (see Celebi, Duran, Tuncel and Akaydin, 2012).
- The last eight matrices in Table 3 are challenging large matrices because they are relatively denser or ill-conditioned.
- The error labelled Error 1 occurs for small number of cores.
- We meet with an error message labelled Error 2 related to buffer size during the factorization subroutine pdgstrf, for the hepta-diagonal matrices.
- Error 3 is a CUDA stream error related to setting cuBLAS library execution stream.

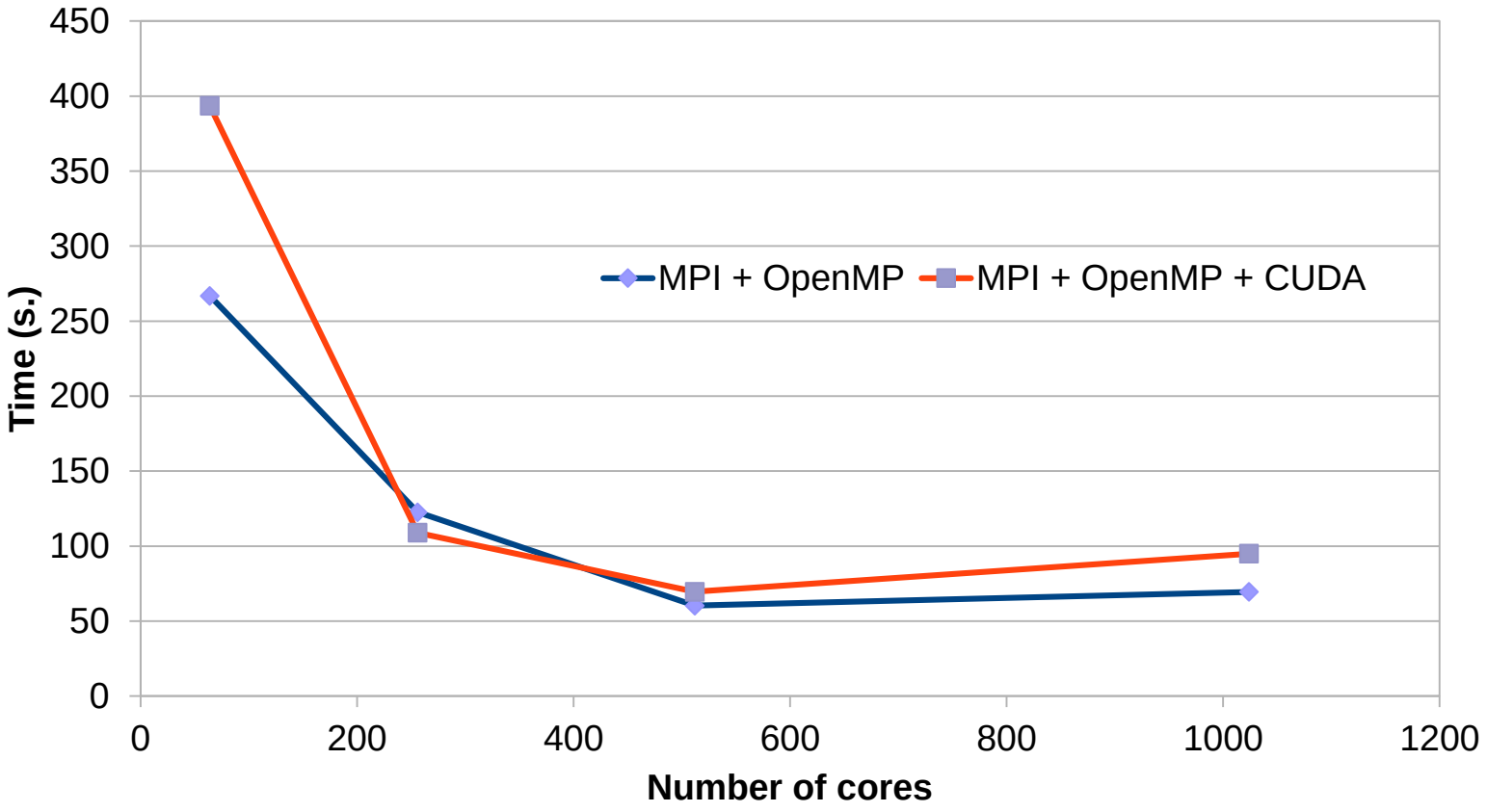
Table II. The configuration of MPI+OpenMP and MPI+OpenMP+CUDA for the direct solver

Testbed:CURIE/	hybrid	hybrid	hybrid	hybrid
SuperLU_DIST version	4	4	4	4
# of cores	64	256	512	1024
# of processes	16	64	128	256
# of threads per process	4	4	4	4
# of GPUs per process	1	1	1	1

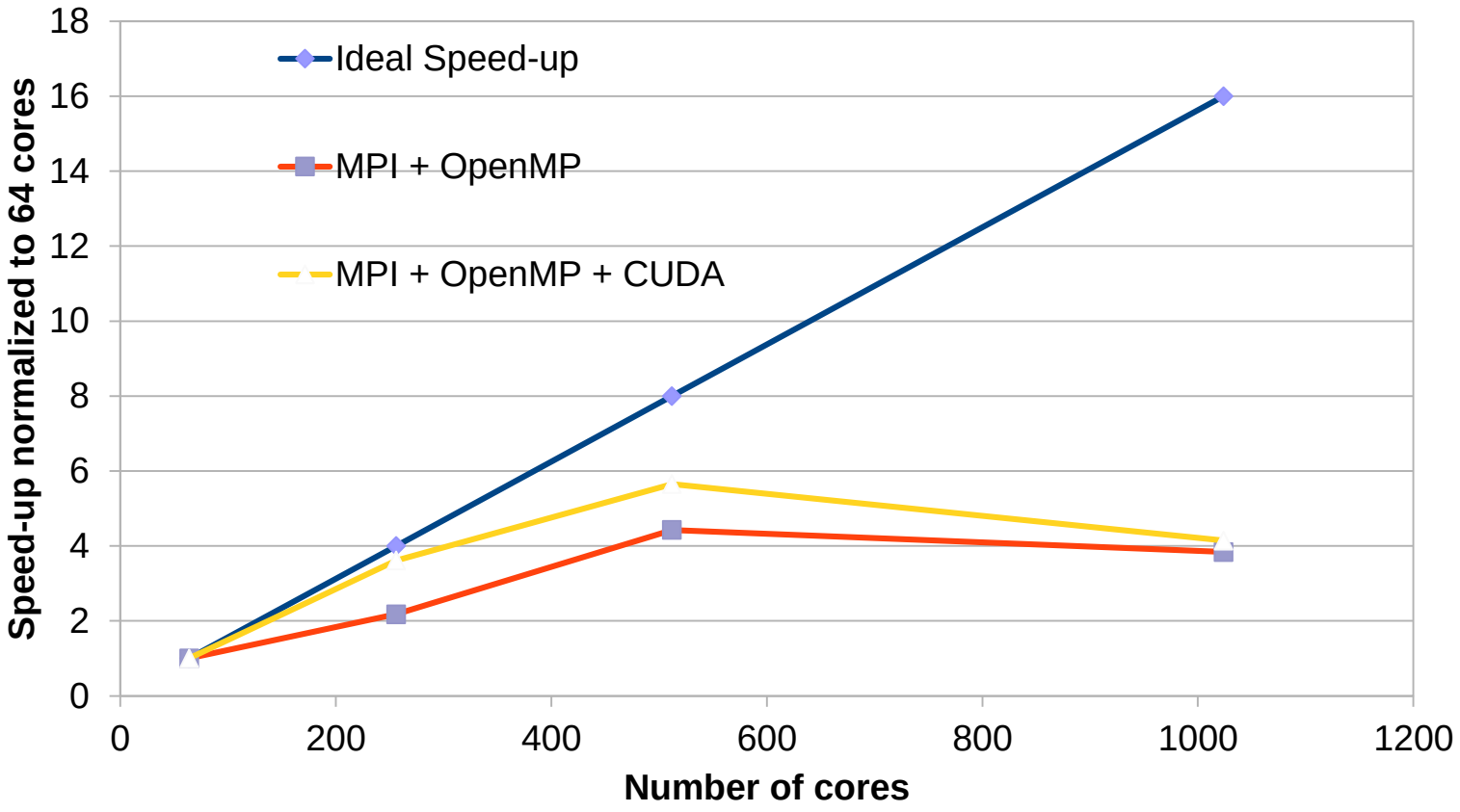
Table III. Wall clock times (s) of SuperLU_DIST 4.0 for the penta-diagonal 2D problems and hepta-diagonal 3D problems on MPI + OpenMP versus MPI + OpenMP + CUDA implementations

Matrices	/ Number of cores	64	256	512	1024
mC_8M	MPI + OpenMP	99.96	34.70	28.78	37.89
	MPI + OpenMP + CUDA	94.70	39.10	43.70	60.72
mC_16M	MPI + OpenMP	230.30	83.19	47.73	59.02
	MPI + OpenMP + CUDA	236.83	87.23	60.00	81.41
mC_6M_D	MPI + OpenMP	Error 1	260.38	296.74	239.52
	MPI + OpenMP + CUDA	Error 1	Error 2	254.44	257.15
mC_8M_D	MPI + OpenMP	Error 1	1005.96	516.86	387.20
	MPI + OpenMP + CUDA	Error 1	680.25	Error 2	353.40
mC_8M_n	MPI + OpenMP	94.70	31.00	32.79	35.83
	MPI + OpenMP + CUDA	70.94	38.27	Error 3	61.34
mC_16M_n	MPI + OpenMP	181.53	75.93	49.53	58.61
	MPI + OpenMP + CUDA	233.22	75.58	61.42	83.61
mC_20M_n	MPI + OpenMP	266.82	122.59	60.30	69.49
	MPI + OpenMP + CUDA	393.49	108.90	69.60	94.99
mC_6M_n_D	MPI + OpenMP	1178.51	409.15	248.84	211.70
	MPI + OpenMP + CUDA	782.22	294.14	Error 2	222.04
mC_8M_n_D	MPI + OpenMP	Error 1	948.03	533.78	386.72
	MPI + OpenMP + CUDA	Error 1	682.02	Error 2	349.16
mC_10M_n_D	MPI + OpenMP	Error 1	877.92	465.60	373.09
	MPI + OpenMP + CUDA	Error 1	752.78	Error 2	Error 3

Wall-clock time of direct solver for mC_20M_n on Curie hybrid nodes



Speed-up of direct solver for mC_20M_n on Curie hybrid nodes



Conclusions

- We compared the CPU performance of the iterative solver icoFoam and the hybrid parallel codes (MPI+OpenMP) of a direct solver SuperLU_DIST 4.0 at TGCC Curie thin nodes at CEA, France.
- We observe that the iterative solver with a diagonal incomplete LU preconditioned bi-conjugate gradient outperforms the direct solver *SuperLU_DIST 4.0* for the simulation matrices.
- We compared the performance of the hybrid parallel codes of MPI+OpenMP+CUDA versus MPI+OpenMP implementation of SuperLU_DIST 4.0 at TGCC Curie hybrid nodes of CPU + GPU at CEA.
- We generally notice that MPI+OpenMP implementation outperforms the hybrid of MPI+OpenMP+CUDA for the set of simulation matrices when we consider the wall clock times for the optimal number of cores.

Conclusions

- There are several overheads coming from CUDA implementation for the complex direct solver algorithm.
- We met with several errors for the challenging simulation matrices.
- We believe that the technology developments in emerging CPU+GPU systems will increase the scalability of related complex algorithms by eliminating the bottlenecks coming from communication and right matching of system components required for special applications

Acknowledgement

- This research was supported by the Project 2010PA2505 awarded under the 18th Call for PRACE Preparatory Access and we acknowledge that the results of this research have been achieved using the PRACE Research Infrastructure resource TGCC Curie (a modern Tier-0 system) based at CEA in France.

References

1. A. Duran, M.S. Celebi, S. Piskin, and M. Tuncel, “Scalability of OpenFOAM for bio-medical flow simulations,” Journal of Supercomputing, 71(3), 2015, pp. 938-951.
2. X. S. Li, J. W. Demmel, J. R. Gilbert, L. Grigori, M. Shao, and I. Yamazaki, SuperLU Users' Guide, Tech. Report UCB, Computer Science Division, University of California, Berkeley, CA, 1999, update: 2011
3. M.S. Celebi, A. Duran, M. Tuncel and B. Akaydın, Scalable and improved SuperLU on GPU for heterogeneous systems, *PRACE (Partnership for Advanced Computing in Europe),* PRACE PN: *283493, PRACE-2IP white paper, Libraries,* WP 44, July 13, 2012.
4. P. Sao, R. Vuduc, and X.S. Li, “A distributed CPU-GPU sparse direct solver,” Euro-Par 2014 Parallel Processing, Lecture Notes in Computer Science vol. 8632, 2014, pp. 487-498.
5. http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm.
6. N. Meyer, M. Lawenda, et al., Best Practices for HPC Procurement and Infrastructure, PRACE-2IP project, under Grant agreement No. RI-283493, Aug. 2013.
7. J. David, JN Richet, E. Boyer, N. Anastopoulos, G Collet, GC Verdiere, et al., Best Practice Guide - Curie v1.17, PRACE, Nov. 2013.
8. TOP500 Supercomputing sites, http://top500.org/
9. The Green500 List, http://www.green500.org
10. OpenFOAM main site. http://www.openfoam.com

QUESTIONS COMMENTS

?

THANK YOU