ANL/ALCF/ESP-13/7

Petascale, Adaptive CFD (ALCF ESP Technical Report)

ALCF-2 Early Science Program Technical Report

Argonne Leadership Computing Facility

ANL/ALCF/ESP-13/7

Petascale, Adaptive CFD (ALCF ESP Technical Report)

ALCF-2 Early Science Program Technical Report

prepared by
Kenneth E. Jansen[1], Michel Rasquin[2]
[1]University of Colorado, Boulder
[2]Argonne Leadership Computing Facility, Argonne National Laboratory

May 7, 2013

ALCF ESP Technical Report

Petascale Adaptive CFD

PI: Kenneth E. Jansen, University of Colorado Boulder
ESP post-doc: Michel Rasquin, Argonne Leadership Computing Facility

March 2013

1 Description of science

The aerodynamic simulations of this project involve modeling of active flow control based on synthetic jet actuation that has been shown experimentally to produce large-scale flow changes (e.g., re-attachment of separated flow or virtual aerodynamic shaping of lifting surfaces) from micro-scale input [1, 2, 3]. This micro-scale input consists for instance in 0.1 W piezoelectric disk which resonate in a cavity alternately and pushes/pulls out/in the fluid through a small slit to create small-scale vortical structures that interact with, and thereby dramatically alter, the cross flow. This process has yet to be understood fundamentally. Synthetic jet actuators offer the prospect of not only augmenting lift but also other forces and moments in a dynamic and controlled fashion for a range of operating conditions. They have been demonstrated to restore and maintain flow attachment and reduce vibrations in wind turbine blades during dynamic pitch, thereby reducing unsteady loads on gearboxes that are currently the prime failure point. In virtual-shape flight control surfaces for aerial vehicles (including commercial airplanes), conventional control surfaces (e.g., flaps, rudder, etc.) can be augmented or even replaced with active flow control, thus improving their lift-to-drag ratio and/or control power. The goal of the numerical simulations proposed in this project is to provide a complementary and detailed view of the flow interactions at a much higher Reynolds number, approaching engineering application scales for the first time.

2 Overview of numerical methods

2.1 Implicit finite element flow solver PHASTA

Flow computations are performed using a CFD flow solver called PHASTA ("Parallel Hierarchic Adaptive Stabilized Transient Analysis"). This code is based on fully-implicit, stabilized, semi-discrete finite element method for the transient, incompressible Navier-Stokes partial differential equation (PDE) governing fluid flows. In particular, we employ the streamline upwind/Petrov-Galerkin (SUPG) stabilization method to discretize the governing equations [4, 6]. The stabilized finite element formulation currently utilized has been shown to be robust, accurate and stable on a variety of flow problems. In our CFD flow solver, the Navier-Stokes equations (conservation of mass, momentum and energy) plus any auxiliary equations (as needed for turbulence models or level sets in two-phase flow) are discretized in space and time. The discretization in space based on a stabilized finite element method leads to a weak form of the governing equations, where the solution (and weight function) are first interpolated using hierarchic, piecewise polynomials, and followed by the computation of integrals appearing in the weak form using Gauss quadrature. Implicit integration in time is then performed using a generalized-α method which is second-order accurate and provides precise control of the temporal damping to reproduce Gears Method, Midpoint Rule, or any blend in between [5]. On a given time step, the resulting non-linear algebraic equations are linearized to yield a system of linear equations $\mathbf{Ax} = \mathbf{b}$ which are solved using Krylov iterative procedures such as GMRES ("Generalized Minimal RESidual" method).

2.1.1 Parallelization

Finite element methods are very well suited for use on parallel computers as substantial computational effort is divided into two main tasks: 1) the calculation of element level integrals leading to the linear system assembly and 2) the solution of the resulting system of algebraic equations $\mathbf{Ax} = \mathbf{b}$. Both of these work types can be equally divided among the processors by partitioning the aggregate mesh into equal load parts [12, 13]. So far, PHASTA is a pure MPI based code and each process executes of copy of the analysis code to handle the computation work and interactions corresponding to its mesh part. Element-based mesh partitioning is currently used for the domain decomposition approach and leads to a natural parallelization for element-integration/equation-formation stage making it highly scalable. In element-based partitioning, each element is uniquely assigned to a single part but in turn leads to shared

dofs or unknowns in the system of equations, which results from the duplicated vertices on the inter-part boundaries. This element-based partitioning along with the control relationships between multiple images of shared vertices is illustrated in Figure 1.

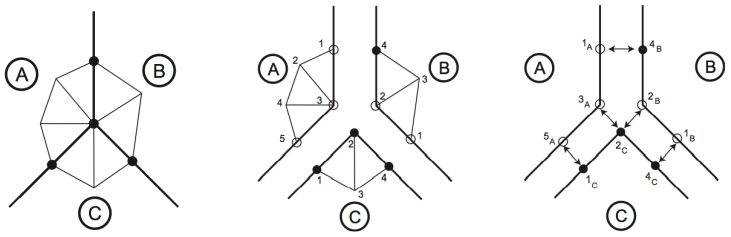
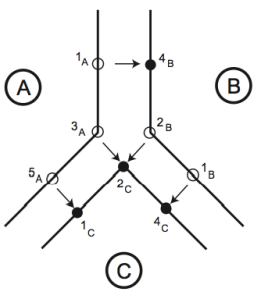


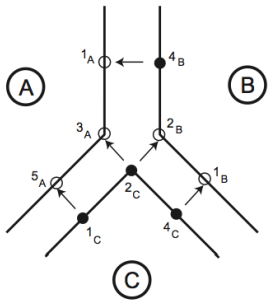
Figure 1: Element-based partitioning and the control relationships between multiple images of duplicated and shared vertices. Solid dot on the middle and right figures denotes an owner image whereas hollow ones indicate non-owners.

Collectively, all the processes have the same information as in the unpartitioned or serial case but no one process holds or has knowledge of the entire tangent matrix, $\mathbf{A}$, nor the residual vector $\mathbf{b}$. Thus, to be able to progress the computations in parallel and march in time, interactions between shared dofs are completed via communications.

After numerical integration on local part (i.e. task 1 mentioned in the previous paragraph for the system assembly), values in rows of $\mathbf{b}$ for shared dofs are individually incomplete on each part (referred to as on-part value) because their contributions are distributed among their images (in finite element methods this is due to the compact support of basis or shape functions used). To obtain a complete value for a vector entry associated with a shared dof, peer-to-peer communications related to shared dofs are required and are implemented in two stages, as illustrated in Figure 2. First, data is accumulated at owner image vertices to obtain complete values. Then, complete values are copied from owners to update their non-owner images. Although one could elect to communicate the on-part entries of the tangent matrix $\mathbf{A}$ to make them complete, our approach does not, limiting communications to vector entries only (such as in $\mathbf{b}$). This matrix update will be achieved implicitly during the resolution of the linear system, as explained in the next paragraph.



(a) Non-owners send while owners receive and accumulate.



(b) Owners send whereas non-owners receive and update.

Figure 2: Communication steps involved to obtain complete values for shared dofs.

The second task of our implicit solve involves finding the solution update vector $\mathbf{x}$. Krylov iterative solution techniques such as GMRES are currently used for that purpose. These techniques employ repeated products of $\mathbf{A}$ with a series of vectors (say, $\mathbf{p}$) to construct an orthonor-

mal basis of vectors used to approximate $\mathbf{x}$. In this series, the outcome of any matrix-vector product is another vector $\mathbf{q}$ ($= \mathbf{Ap}$) which is used to derive the subsequent vector in the series. The first vector in this series is derived from the residual vector $\mathbf{b}$ which contains complete values at this point of the solve. Even though $\mathbf{A}$ contains only on-part values for shared dofs and is incomplete, it is still possible to perform the basic kernel of (sparse) matrix-vector product, i.e., $\mathbf{q} = \mathbf{Ap}$, provided vector $\mathbf{p}$ has been first updated in the same way as illustrated in Figure 2 for vector $\mathbf{b}$ and contains complete values. Due to the distributive property of $\mathbf{Ap}$ product, the resulting vector $\mathbf{q}$ will in turn contain on-part (incomplete) values. Therefore, after a local $\mathbf{Ap}$ product, our two-pass communication strategy illustrated in Figure 2 must be applied again to obtain complete values in $\mathbf{q}$ (provided $\mathbf{p}$ contained already complete values). Before proceeding to the next product in the series, it is important to note that computation of norms is required to perform orthonormalization. In this step, the norm of vector $\mathbf{q}$, and its dot-product with the previous vectors in the series, are computed. To compute a norm or dot-product, first a local dot-product is computed (requiring no communication) but then, to obtain a complete dot-product, a sum across all cores is needed. A collective communication (of allreduce type) is used to carry out the global summation. To summarize task 2, successive $\mathbf{Ap}$ products are carried out along with peer-to-peer communications to obtain complete values and with global communications to perform the orthonormalization. This series of steps leads to an orthonormal basis of vectors which is used to find an approximate update vector $\mathbf{x}$ and marks the end of a non-linear iteration step.

2.2 Adaptive mesh control and mesh partitioning

In addition to the CFD flow solver PHASTA, adaptive meshing [8, 9, 11, 10] and mesh partitioning [7] techniques are other essential ingredients required to generate and partition significantly large (in the order of 5 billion or more elements) 3D unstructured finite element meshes for the target applications. Indeed, the application of reliable numerical simulations requires them to be executed in an automated manner with explicit control of the approximations made. Since there are no reliable a priori methods to control the approximation errors, adaptive methods must be applied where the mesh resolution is determined in a local fashion based on the spatial distribution of the solution and errors associated with its numerical approximation. Furthermore, the reliability and accuracy of simulations is also a strong function of the mesh quality and configuration. In many physical problems of interest, especially in the field of fluid mechanics, solution features are most effectively resolved using mesh elements which are oriented and configured in a certain manner. For example, in the case of viscous flows, use of boundary layer meshes is central to the ability to effectively perform the flow simulations due to their favorable attributes, i.e., high-aspect ratio, orthogonal, layered and graded elements near the viscous walls. As shown in Section 5, while the quality of the mesh is essential for the reliability and the quality of the solution, the quality of the mesh partition plays a key role in the scalability of the flow solver.

3 Problem size

The usual productive runs on Intrepid concerned cases that included up to O(0.5 billion) finite elements. On Mira, the new available resource allow us to consider typical problems that include O(5 billion) finite elements, that is a factor 10 increase w.r.t. Intrepid. This factor will allow us to increase the Reynolds number of our simulations and approach engineering application scales for the first time.

4 Codes and packages involved in this project

The execution of the flow solver PHASTA and the adaptive meshing and mesh partitioning procedures rely on the following third-party libraries and softwares: MPI, Zoltan, ParMETIS, IPComMan, PIMA, FMDB, phParAdapt, ParMa, Simmetix, ACUSIM and Parasolid/Acis.

5 Performance on Mira

The performance of PHASTA on MIRA is illustrated in Figure 3 and Table 1 with a strong scaling study. These simulations associated with a flow control application are performed on a 3.3 billion finite element mesh. The number of MIRA nodes in this scaling study ranges from 2,048 to 32,768 (corresponding to respectively 16,384 and 524,288 cores) and the number of MPI processes per core varies from 1 to 4. The simulation with 2,048 nodes (32,768 cores) and 1 MPI process per core is considered as the reference hereafter.

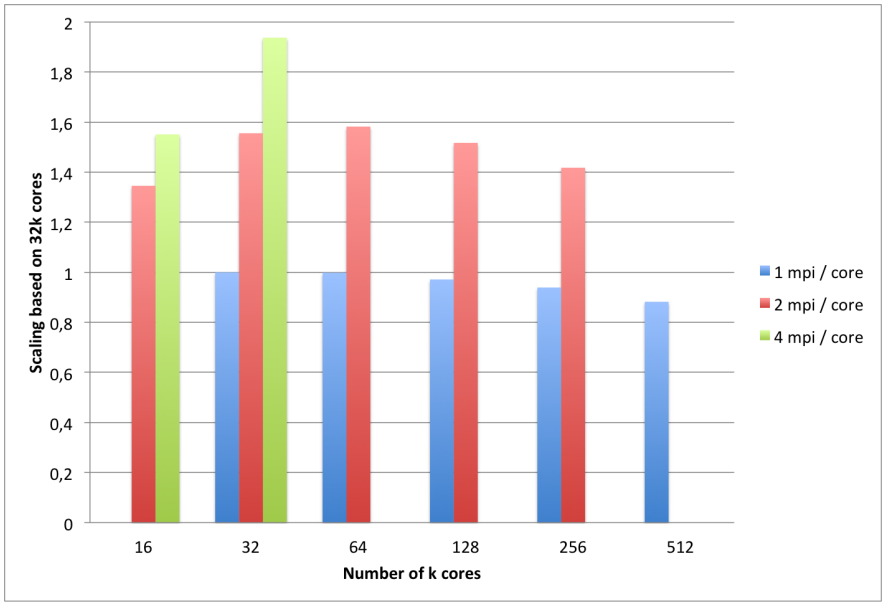


Figure 3: Strong scaling performance for PHASTA on MIRA for a 3.3 billion finite element mesh.

First, a consistent scaling factor of 88.22% is observed on 32,768 nodes (524,288 cores) with one MPI process per core w.r.t. to our reference. Increasing the number of MPI processes per core to 2 or 4 leads to an additional significant improvement of the performance. Indeed, 16,384 nodes (262,144 cores) with one MPI process per core leads to a scaling factor of 92,92%, whereas 2 MPI processes per core leads to a scaling factor of 141,78%. On 2,048 nodes (32,768 cores), 4 MPI processes per core leads to a scaling factor of 193.73%, 2 MPI processes to 155.56% and 1 MPI process per core to 100% (reference case). Finally, MIRA appears to be about 11 times faster than Intrepid on a node basis when 1 MPI process per core is considered on both MIRA and Intrepid. When 2 MPI processes per core are considered on MIRA, the acceleration factor per node rises up to about 18 (with still the standard 1 MPI process per core on Intrepid in this comparison).

As mentioned in the previous section, the scaling of our flow solver strongly depends on the quality of the mesh partitioning. Indeed, one important point to consider during the partitioning of the mesh is that the computational load (in any part) during the system formation stage (i.e., during formation of the tangent matrix $\mathbf{A}$ and residual vector $\mathbf{b}$) depends on the number of elements present in the part, whereas the system solution stage depends on the degrees-of-freedom (dofs) or unknowns in the system of equations on that part, which is proportional to the number of vertices. Typically, element balance (with sufficient load per part) results in a reasonable dof balance as well. As long as the dof balance is preserved, the element-based partitioning also maintains the scalability in the iterative linear solve step, as illustrated in

K Nodes	K Cores	MPI process per core = part/core	K mesh parts = MPI process	Wall Clock (s)	Scaling factor (%) Ref: 2K nodes BG/Q with 1 mpi/core	Mira/Intrepid node factor Ref: 40K nodes BG/P with 1 mpi/core
2 (ref)	32	1	32	206.09	100.00	12.04
4	64	1	64	103.21	99.84	12.02
8	128	1	128	53.04	97.14	11.69
16	256	1	256	27.43	93.92	11.30
32	512	1	512	14.6	88.22	10.62
1	16	2	32	306.31	134.56	16.20
2	32	2	64	132.48	155.56	18.72
4	64	2	128	65.13	158.21	19.04
8	128	2	256	33.96	151.72	18.26
16	256	2	512	18.17	141.78	17.06
1	16	4	64	265.77	155.09	18.67
2	32	4	128	106.38	193.73	23.32

Table 1: Strong scaling results for PHASTA on MIRA with a 3.3 billion finite element mesh.

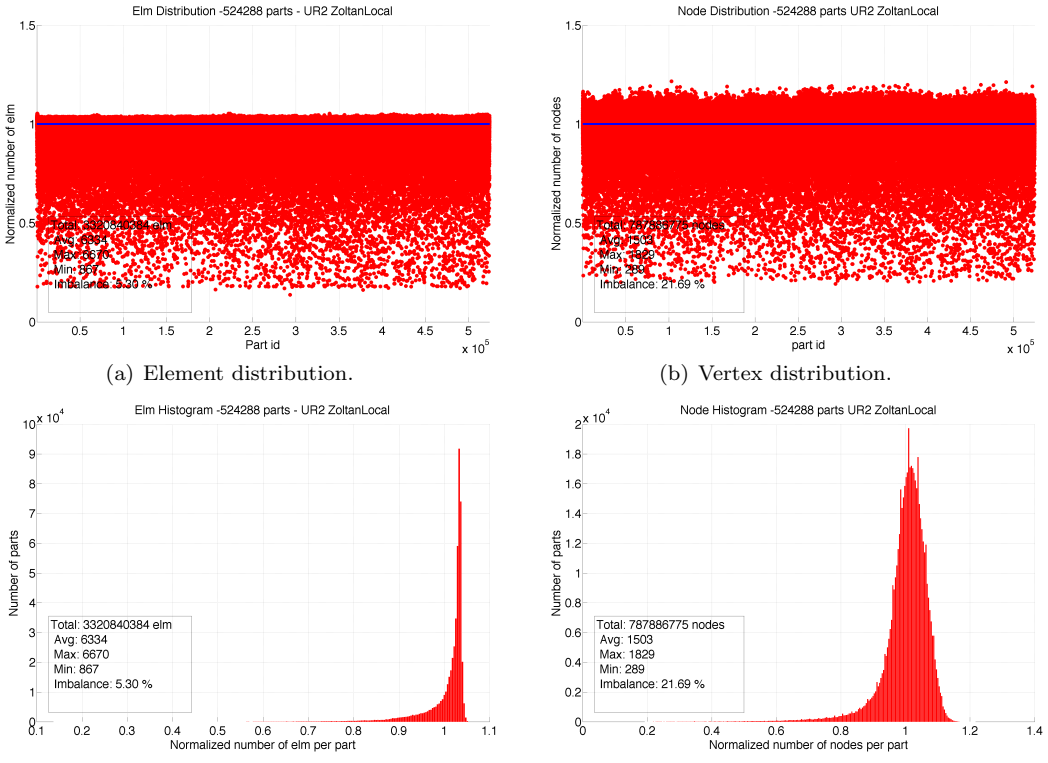
Figure 3 with the simulations performed with 1 MPI process per core.

For a mesh with fixed element topology (e.g. tetrahedra), balanced parts within a partition imply that each part contains as close to the average number of mesh entities (both elements and vertices) as possible. However, in situations where the number of mesh elements per part is relatively small (in the order of few thousand), significant imbalance in dofs can result while the element imbalance remains under control. Indeed, the balance of dofs is not explicitly requested by pure element-base partitioners. This dofs imbalance increase is illustrated in Table 2 and highlighted in particular with the mesh partitioned in 524,288 and 1,572,864 parts. Furthermore, the percentage of shared dofs on part boundaries increases in situations where a fixed-size problem is spread over more and more parts as is the case during this strong scaling study and thus, eventually becomes detrimental to scaling. Finally, it is also common for pure element-based partitioners to generate empty parts in such extreme conditions that flow solvers typically cannot handle.

To illustrate further this dof imbalance as the the mesh is partitioned in more and more parts with a pure element-base partitioner, the mesh entity distribution per part and associated histogram are presented in Figures 4 and 5 for respectively 524,288 and 1,572,864 parts. Figures 4(a) and 4(c) show that most of the parts are well balanced in terms of element. The element histogram also confirms that the node distribution is relatively smooth, as most parts present a number of elements above the target average by only 5.30% at most. However, the conclusion is different for the vertex distribution and histogram presented in Figures 4(b) and 4(d). Indeed, the node distribution is characterized by a number of heavily loaded parts (based on mesh vertices) referred to as spikes, with a number of vertices significantly above above the target average. These spikes are significant contributors to the degradation of the scaling of the iterative linear solver. Moreover, the right tail of the node histogram shows that only a few fraction of the parts are too heavily loaded w.r.t. the target average. This tendency is emphasized in Figure 5, where the element-base partitioning of the mesh in 1,572,864 parts also generates a few empty parts that solvers typically cannot handle. Solutions to remedy to these problems are presented in Section 6 and are currently being implemented.

Parts	32K		64K		128K	
	Elements	Vertices	Elements	Vertices	Elements	Vertices
Total	3.3 billion	628,518,185	cst	658,364,797	cst	690,969,857
Target avg	101,344	19,181	50,672	10,046	25,336	5,272
Max in part	107,087	20,910	53,362	11,053	26,681	5,940
% imbalance	**5.67**	**9.01**	**5.31**	**10.02**	**5.31**	**12.67**
Min in part	56,203	10,790	28,427	5,824	11,993	2,610
Parts	256K		512K		1536K	
	Elements	Vertices	Elements	Vertices	Elements	Vertices
Total	cst	733,192,244	cst	787,886,775	cst	905,248,408
Target avg	12,668	2,797	6,334	1,503	2,111	576
Max in a part	13,340	3,267	6,670	1,829	2,231	875
% imbalance	**5.30**	**16.80**	**5.30**	**21.69**	**5.68**	**51.91**
Min in a part	5,251	1,170	867	289	0	0

Table 2: Element and vertex imbalance for a 3.3 billion finite element mesh



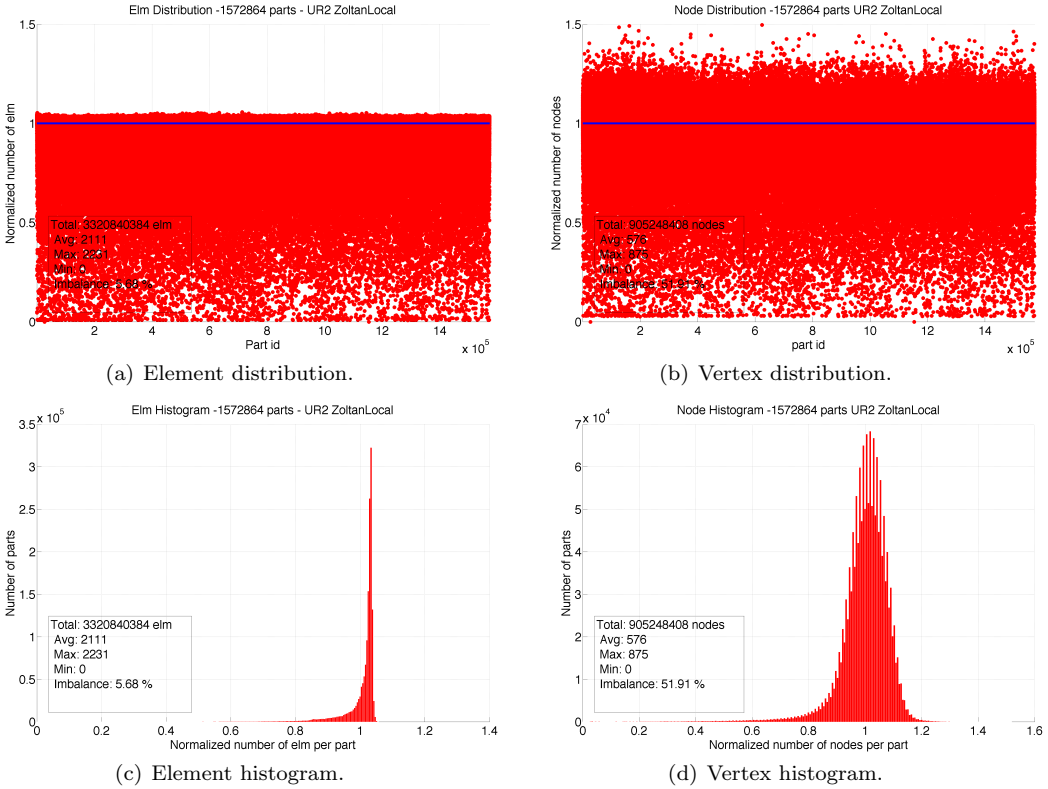
(a) Element distribution.

(b) Vertex distribution.

(c) Element histogram.

(d) Vertex histogram.

Figure 4: Mesh entity distribution per part (top) and histogram (bottom) for a 3.3 billion element mesh partitioned in 524,288 parts with Zoltan and ParMetis. The blue line in the distribution plots represents the target average number of entities per part.



(a) Element distribution.

(b) Vertex distribution.

(c) Element histogram.

(d) Vertex histogram.

Figure 5: Mesh entity distribution per part (top) and histogram (bottom) for a 3.3 billion element mesh partitioned in 1535K parts with Zoltan and ParMetis. The blue line in the distribution plots represents the target average number of entities per part.

6 New algorithms for Mira

6.1 Unstructured mesh partitioning at large scale

Two algorithms aiming at improving the quality of the mesh partitioning are being implemented in a library called Parma, in coordination with the Scientific Computation Research Center at the Rensselaer Polytechnic Institute (RPI). The first algorithm whose first version was presented in [13] consists in migrating locally a small number of elements from heavily loaded parts to relatively lightly loaded neighboring parts in order to reduce the vertex imbalance without perturbing much of the element balance. The result of this algorithm is illustrated in Table 3 and Figures 6 and 7 with an extreme example that includes 180 million elements partitioned in 131,072 parts.

Parts	128K (no Parma)		128K (with Parma)	
	Elements	Vertices	Elements	Vertices
Total	180 million	54,043,396	cst	52,723,643
Target avg	1,378	412	1,378	402
Max in a part	1,455	564	1,604	462
% imbalance	**5.59**	**36.89**	**16.40**	**14.93**
Min in a part	0	0	1	4

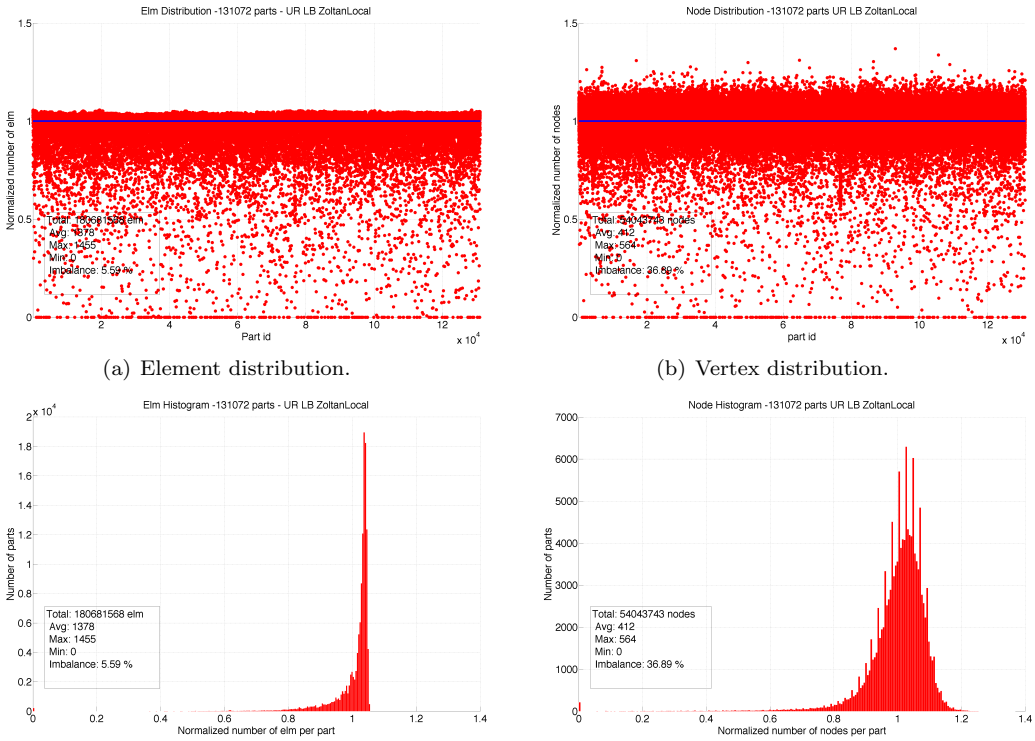
Table 3: Element and vertex imbalance for a 180 million element mesh partitioned in 131,072 parts with Zoltan and ParMetis, and improved with Parma.

From this table and associated figures, one can observe that the vertex imbalance is significantly improved from 36.89% to 14.93%. In this extreme example, the imbalance for both vertices and elements reaches a similar level after this first operation. Migrating a few elements locally from heavily loaded parts to lightly loaded neighboring parts not only reduces the vertex imbalance but also usually decreases the total number of vertices on part boundaries which in turn decreases the total inter part communication. As a first simple solution to fix empty parts, one single element is also migrated from a neighboring part to an empty part.

However, this first algorithm fails at removing all the spikes in the vertex distribution, especially when heavy loaded parts are concentrated in the same region of the mesh. A second algorithm referred to as heavy part splitting approach is currently investigated for that purpose. This algorithm is applied after the smooth element migration described in the previous paragraph and includes three steps. The first step consists in estimating the number of heavy loaded parts in the mesh (both in terms of vertex and elements). The second step consists in generating as many empty parts as needed by grouping lightly loaded parts together. The empty parts potentially generated by the pure element-base partitioning are also recycled for that purpose. Finally, the third step consists in splitting the heavy loaded parts in two or more parts and migrating these new split parts to the empty parts. These heavy loaded parts are split based on their geometry in such a way that the total number of new shared vertices is minimized.

6.2 Live and interactive data co-visualization

Fully implicit finite element methods applied to Computational Fluid Dynamics have already been demonstrated to scale very well up to hundred of thousands of processors, provided that the load in terms of both element and vertex per processor is carefully balanced for respectively the equations formation and iterative solve. This scalability of the flow solver is essential for simulation of turbulent flows that requires the Navier-Stokes equations to be solved on highly refined meshes over a considerable number of small time steps. However, it may take orders of magnitude longer time to perform any reasonable assessment of the insight gained due to the



(a) Element distribution.

(b) Vertex distribution.

(c) Element histogram.

(d) Vertex histogram.

Figure 6: Mesh entity distribution and histogram among the mesh parts for a 180M element mesh partitioned in 131,072 parts with Zoltan and ParMetis but without Parma. The blue line in the distribution plots represents the target average number of entities per part.

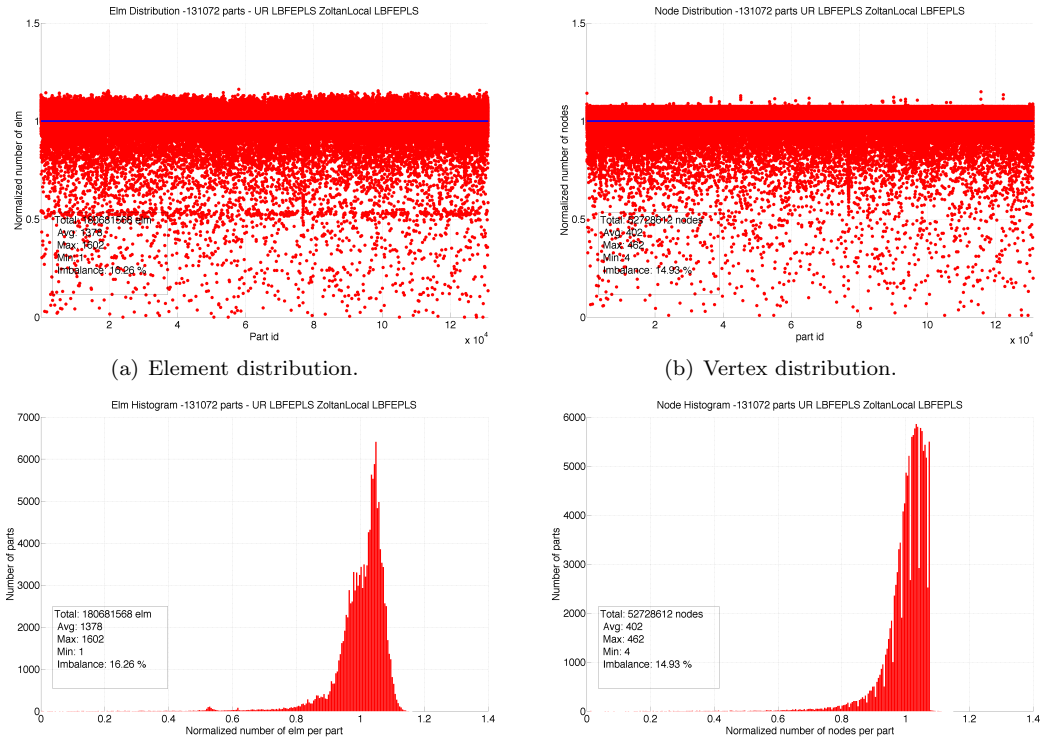
time it takes to write the data, load the data into post processing software, and to analyze and display insightful results. In coordination with the University of Colorado at Boulder, Kitware, INC, the Rensselaer Polytechnic Institute and ALCF, we now consider a more strict definition of "solution" whereby a live data analysis is able to provide continuous and reconfigurable insight into massively parallel simulations, paving the way for interactive simulation and simulation steering. Specifically, we demonstrated our co-visualization concept of either the full data or in situ data extracts on 163,840 cores of the Blue Gene/P Intrepid system tightly linked through a high-speed network to 100 visualization nodes of the Eureka system that share 800 cores and 200 GPUs, as illustrated in Figure 8. In particular, we used this technique to visualize vortical structures that arise from the interaction of a cross flow with an array of synthetic jets on a realistic wing with application to flow control. We plan to port this capability to Mira and Tukey soon.

References

[1] M. Amitay, B.L. Smith, and A. Glezer, *Aerodynamic Flow Control Using Synthetic Jet Technology*, AIAA Paper, 208, 1998

[2] A. Glezer and M. Amitay, *Synthetic Jets*, Annual Review of Fluid Mechanics, 34(1):503-529, 2002.

[3] O. Sahni, J. Wood, K.E. Jansen, and M. Amitay, *Three-dimensional interactions between a finite-span synthetic jet and a crossflow*, Journal of Fluid Mechanics, 671:254-287, 2011.



(a) Element distribution.

(b) Vertex distribution.

(c) Element histogram.

(d) Vertex histogram.

Figure 7: Mesh entity distribution and histogram among the mesh parts for a 180M element mesh partitioned in 131,072 parts with Zoltan and ParMetis, and improved with Parma. The blue line in the distribution plots represents the target average number of entities per part.

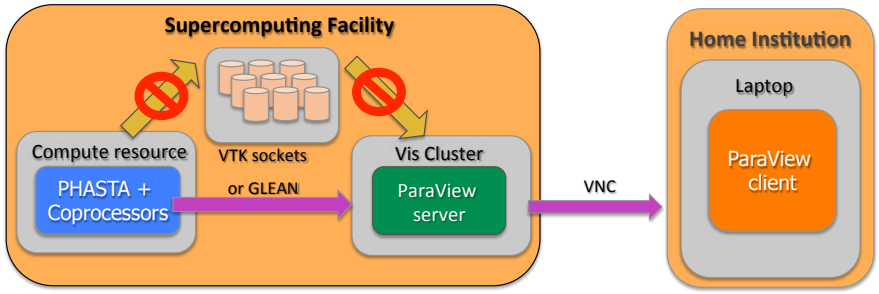


Figure 8: Workflow for live co-visualization of data.

[4] C.H. Whiting and K.E. Jansen *A stabilized finite element method for the incompressible Navier-Stokes equations using a hierarchical basis*, International Journal of Numerical Methods in Fluids, 35:93-116, 2001.

[5] K.E. Jansen, C.H. Whiting and G.M. Hulbert, *A generalized-α method for integrating the filtered Navier-Stokes equations with a stabilized finite element method*, Computer Methods in Applied Mechanics and Engineering, 190:305-319, 1999.

[6] A. K. Karanam, K. E. Jansen, and C. H. Whiting, *Geometry based pre-processor for par-*

allel fluid dynamic simulations using a hierarchical basis, Engineering with Computers, 24(1):1726, 2008.

[7] M. Zhou, O. Sahni, K.D. Devine, M.S. Shephard and K.E. Jansen, *Controlling unstructured mesh partitions for massively parallel simulations*, SIAM Journal of Scientific Computing, 32:3201-3227, 2010

[8] J. Mueller, O. Sahni, X. Li, K.E. Jansen, M.S. Shephard, and C.A. Taylor, *Anisotropic adaptive finite element method for modeling blood flow*, Computer Methods in Biomechanics and Biomedical Engineering, 8(5):295-305, 2005.

[9] O. Sahni, K.E. Jansen, M.S. Shephard, C.A. Taylor, and M.W. Beall, *Adaptive boundary layer meshing for viscous flow simulations*, Engineering with Computers, 24(3):267285, 2008.

[10] M.S. Shephard, K.E. Jansen, O. Sahni, and L.A. Diachin, *Parallel adaptive simulations on unstructured meshes*, Journal of Physics: Conference Series, 78-012053:012053, 2007.

[11] O. Sahni, J. Mueller, K.E. Jansen, M.S. Shephard, and C.A. Taylor, *Efficient anisotropic adaptive discretization of cardiovascular system*, Computer Methods in Applied Mechanics and Engineering, 195(41-43):56345655, 2006.

[12] O. Sahni, M. Zhou, M.S. Shephard, and K.E. Jansen, *Scalable implicit finite element solver for massively parallel processing with demonstration to 160K cores*, Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, 2009, 68:1-68:12.

[13] M. Zhou, O. Sahni, T. Xie, M.S. Shephard and K.E. Jansen, *Unstructured mesh partition improvement for implicit finite element at extreme scale*, The Journal of Supercomputing, 1-11, 2012.

Argonne Leadership Computing Facility
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 240
Argonne, IL 60439

www.anl.gov