44th AIAA Aerospace Sciences Meeting & Exhibit, 9-12 January 2006, Reno, NV

AIAA-2006-0942

Towards a fully parallel integrated geometry kernel, mesh generator, flow solver & post-processor

W N Dawes,[*]
CFD Laboratory, Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK.

This paper reports progress towards the integration of a Level Set based geometry kernel with an octree-based cut-Cartesian mesh generator, RANS flow solver and post-processing within a single piece of software. Recent research has concentrated on parallelizing the entire process with commodity PC clusters as the target. The aim of this work is to eliminate *all* serial bottlenecks from the CFD Process – including the geometry kernel and its editing & management during design optimization. The focus in this particular paper is the mesh generation. The ultimate aim is to allow rapid prototyping design optimization to take place on geometries of arbitrary size in a spirit of a real time computer game.

I. Introduction

Commodity PC clusters are cheap & commonplace and the technical quality of both mesh generators & flow solvers is sufficient for serious, engineering application. In the real world, CFD simulations are getting bigger. In turbomachinery, flutter & noise simulations routinely consider several blade rows covering the whole annulus and demand meshes in the 50M range. Engine-airframe integration simulations have to consider not just whole annuli but also significant local airframe scenery and are already pushing towards 100M cells. Airframe simulations, especially those with deployed flaps or undercarriage, are similarly nudging 100M+ cells. Motorsport simulations are right out in front, especially open wheel designs, with whole car simulations demanding meshes pushing towards 300M. Simulations of pollutant dispersion at the micro-to-meso scale in realistic city geometries look set for 1B cell resolution. So what are the bottlenecks which might inhibit this trend; in particular, where are the *serial* bottlenecks?

Over the years the flow solver itself was considered to be the bottleneck and virtually all parallelization strategies and techniques were devised for and applied to the solver. These days all front rank solvers have good parallel performance and many can sustain that performance on meshes well into the 100M+ range. Curiously, however, most of this work has been "vanilla" in the sense that existing solver technology was simply ported into parallel; the inverse question, what might be a good solver algorithm for efficient parallelization, seems only to have been addressed rather superficially by, for example, choosing explicit rather than implicit matrix-based methods from the existing tried & tested armoury.

Anyone with real-world experience of large simulations quickly realizes that generating & partitioning the mesh represents a key serial bottleneck. The move towards 64 bit architectures will obviously permit bigger meshes to be generated on a given PC but the *speed* of the generation will become unacceptably slow (64 bit machines are little different in speed to 32 bit ones but simply overcome the 4Mb address limit of the latter). There is some research on parallel mesh generation but it is not obvious how to partition efficiently the tried & tested "vanilla" approaches of multi-block structured or Delaunay-based unstructured. Certainly little is available commercially and current limits are of order 60M cells. Most approaches to then build bigger meshes are like "chunking" whereby the CAD model is cut up into smaller pieces (assuming the availability of suitable tools) and then meshed separately with common

[*] wnd@eng.cam.ac.uk

inter-domain boundaries to be merged later (but achieving contiguous viscous layers rapidly become a nightmare…).

Similar issues of size and speed make post-processing a serial bottleneck which cannot be resolved with the shift to 64 bit architectures. One of the very few available parallel post-processing packages is the pioneering pV3 (Haimes [1997]) but even this is limited by rendering rate (the numbers of triangles per second which the graphics engine can comfortable handle) and would need parallel decimation strategies to be devised and implemented.

Then there is the CAD model itself. It would appear that the current geometry modeling paradigm, based on BREP patches/edges/topology bindings, is inherently serial. Why does this matter? It matters because geometries are getting bigger and more complex (and 64 bit will increase model size but with little improvement in processing speed) and also because the trend in automatic design optimization is to interact directly with the CAD. Software like CAPRI (Haimes et al [1999]) enables direct interrogation of the solid modeling kernel underpinning the (hopefully parameterised) CAD model but for large simulations and/or extensive optimizations everything must run in parallel – and one figure of merit for the optimizer is to minimize the number of licences tied up.

The research reported in this paper was motivated by asking the question: what would it take for the *entire* CFD Process from CAD-to-mesh-to-solver-to-post-processing to be inherently parallel, scalable and without any serial bottlenecks? An interesting way forward appears to be the integration of a geometry kernel based on a Level Set approach with an octree-based cut-Cartesian mesh generator, RANS flow solver and post-processing within a single piece of software. Some basic building-block work was reported by Dawes [2005]. More recent research, reported in this paper, has concentrated on parallelizing the entire process with commodity PC clusters as the target. The ultimate aim of this research is to allow rapid prototyping design optimization to take place on real geometries of arbitrary size in a spirit of a real time computer game.

II. An Integrated, Parallel, Geometry Engine, Mesh Generator, Flow Solver & Post-Processor

A. Algorithmic architecture

The background to the new work presented here is the exploration of the possibilities offered by the integration of the solid modelling directly with the mesh generation & with the flow solution. This research combines ideas from solid modelling (see for example Samareh [1999,2001] & Haimes et al [1998]) with virtual sculpting (see for example, Galyean et al [1991], Perng et al [2001] and Baeretzen [2001]) combined in the context of a simple, cut-Cartesian mesh flow solver (see Bussoletti et al [1985] or Aftosmis et al [1998]). A recent publication (Dawes [2005]) set out these building blocks and showed their potential as a rapid prototyping design tool.

The core of the new code ("BOXER") is a very efficient octree data-structure acting *simultaneously* as a search engine, as a spatial occupancy solid model and as an adaptive, unstructured mesh for the flow solver. This provides unlimited geometric flexibility and very robust mesh generation. The solid model is initialised by the import of a tessellated surface from a variety of potential sources (most CAD engines have an STL export) or by direct interrogation of the CAD solid model kernel itself. The solid model is captured on the adaptive, unstructured Cartesian hexahedral mesh very efficiently by cutting the tesselated boundaries using basic computer graphics constructs developed for interactive 3D gaming (aimed at real-time collision detection). This geometry capture is very fast; for example, a body represented by about 1M surface triangles can be imported into a mesh of around 11M cells (with 6-7 levels of refinement) in approximately 2 minutes on a single, top-end PC. The spatial occupancy solid model is then sampled as a distance field and managed as a Level Set; this forms a solid modeling kernel to support the activities of the code. Adaptive mesh refine/de-refine for the flow and for the geometry, via the distance field, enables both moving bodies and topology editing – *flow sculpting* (see Dawes [2005]). The algorithmic architecture is illustrated in Figure 1.

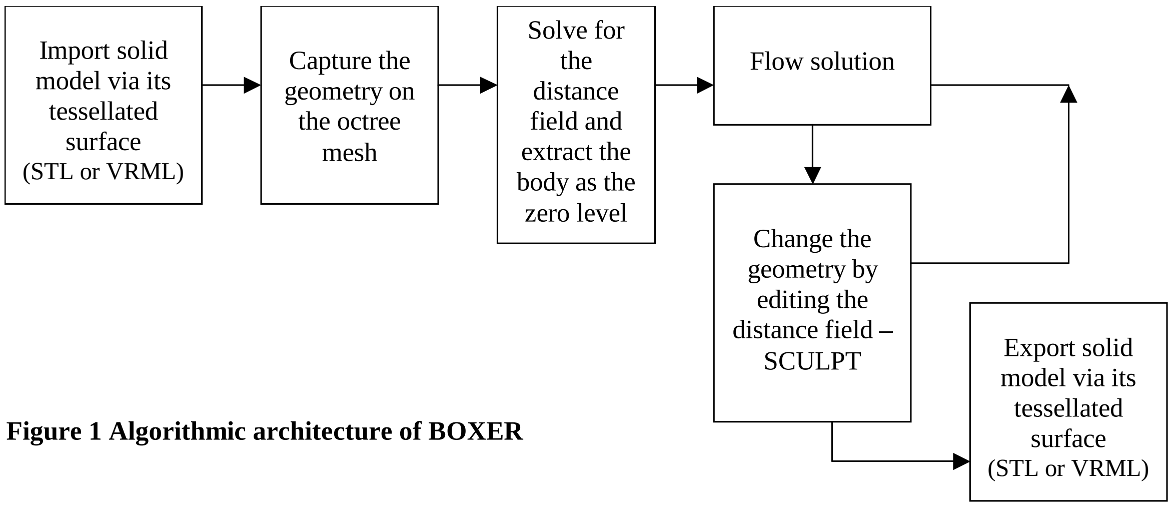

Import solid model via its tessellated surface (STL or VRML)
Capture the geometry on the octree mesh
Solve for the distance field and extract the body as the zero level
Flow solution
Change the geometry by editing the distance field – SCULPT
Export solid model via its tessellated surface (STL or VRML)

Figure 1 Algorithmic architecture of BOXER

The associated 3D RANS solver was adapted from an existing unstructured mesh RANS solver (Dawes et al [2001]) with the additional complications of handling hanging nodes and the cut cells. The cut cells are the big disadvantage of the Cartesian approach. For the present, rather exploratory research, the simple ghost cell approach of Viecelli [1971] was adopted as it is very robust; this relies on storing for each cut cell the local body normal. More accurate approaches are certainly possible and, although not shown in this paper, BOXER has also the ability to delete out the cut cells and build layered viscous meshes along body normals.

B. Software architecture

The algorithmic architecture described above was chosen precisely because each building block is inherently parallelizable.

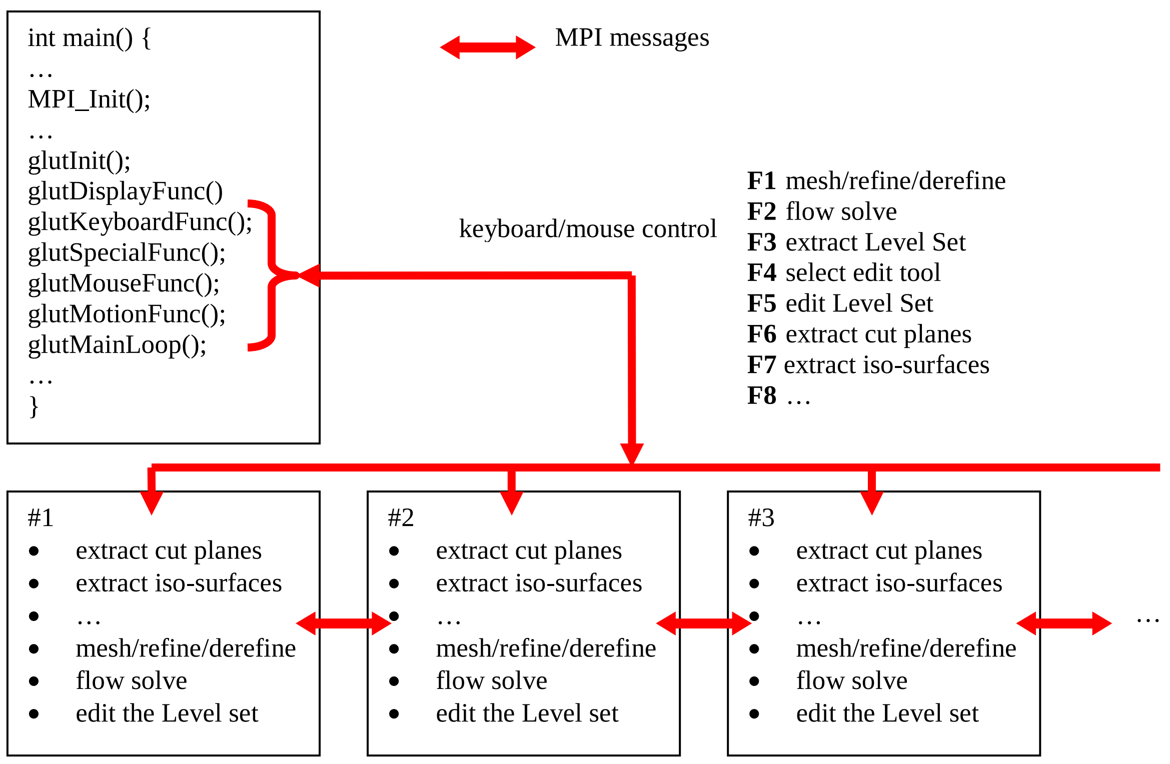
The octree-based cut-Cartesian mesh generation algorithm works very nicely in parallel. The hexahedral cells are cut by local interactions with individual triangles from the STL/VRML input, or with reference to the local value of the distance field. This can be partitioned easily over a number of domains (as can the STL/VRML input itself which may contain millions of triangles). At the moment load balancing between domains is static and based on the spatial distribution of the input geometry triangulation; the next challenge is to introduce dynamic balancing. Communication between domains is relatively small and mainly related to the constraint to maintain cell-cell transitions at no more than h-2h.

The flow solver is straightforward to make parallel and a simple halo cell methodology was implemented as special ghost cells within the code's existing data structures.

Post-processing is relatively easy to implement in parallel. Unlike streamlines, cut surface extraction & iso-surface extraction, for example, do not even need inter-domain communication. Each partition contributes a set of triangles (with each node storing X,Y,Z and a scalar, like Mach number) which is then sent to the master process, combined with all the other contributions, and then rendered subject to selected transformations and lighting models. In future work, the child processes will also decimate their contributions to maximize the front end rendering rate.

The underpinning solid modeling kernel, the Level Set representation of the spatial occupancy solid model, is simply based on manipulating a field variable, the distance field, and so can *also* be trivially parallelized – as can any tools to manipulate the geometry. This appears to offer this sort of solid modeling kernel a big advantage & opportunity compared to the conventional BREP alternatives.

The structure developed to integrate these building blocks in parallel was an event queue managed, child-parent-sibling hierarchy implemented in a mixture of C and FORTRAN using OpenGL/glut and MPI. This architecture is illustrated in Figure 2.



```
int main() {
…
MPI_Init();
…
glutInit();
glutDisplayFunc()
glutKeyboardFunc();
glutSpecialFunc();
glutMouseFunc();
glutMotionFunc();
glutMainLoop();
…
}
```

Figure 2 Software architecture of BOXER

III. Example applications

The focus of this paper is on mesh generation; future publications will be more concerned with flow solving & post-processing. This section contains some sample applications selected to illustrate not only the parallel performance of the mesh generation but also the generation of meshes simply too large for single PC.

A. Classic speed-up tests for a simple shrouded turbine – 2.27M cells

The first example is the simple case of a shrouded turbine blade which can be adequately resolved with a relatively coarse mesh of about 2.27M cells. The hardware deployed was a simple laptop front-end connected via the office network to a cluster of 16 PC's. Each of these PC's has a 2.5GHz Pentium IV processor with only 1Gb RAM; they are interconnected by simple 1Gb/s Ethernet in a dedicated circuit. Figure 3 shows the mesh generation time in seconds on 1, 2, 4, 8 & 16 PC's together with a reference "theoretical" performance. As can be seen, for this small mesh the generation time becomes communication-bound beyond 8 PC's. Nevertheless, 10 seconds is a remarkably small time for a mesh for this class of geometry.

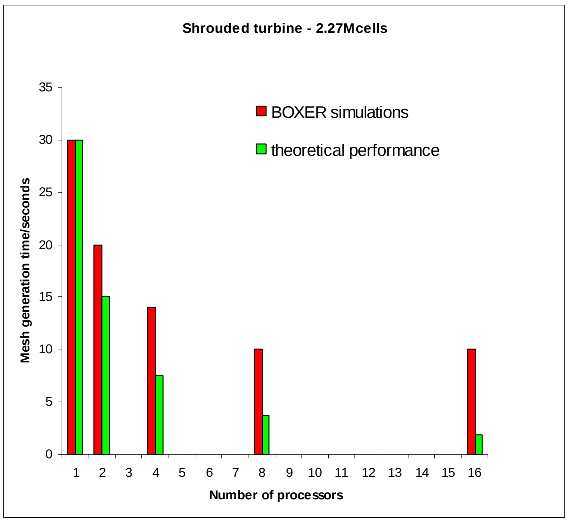


Figure 3 Mesh generation timings for the generic shrouded turbine

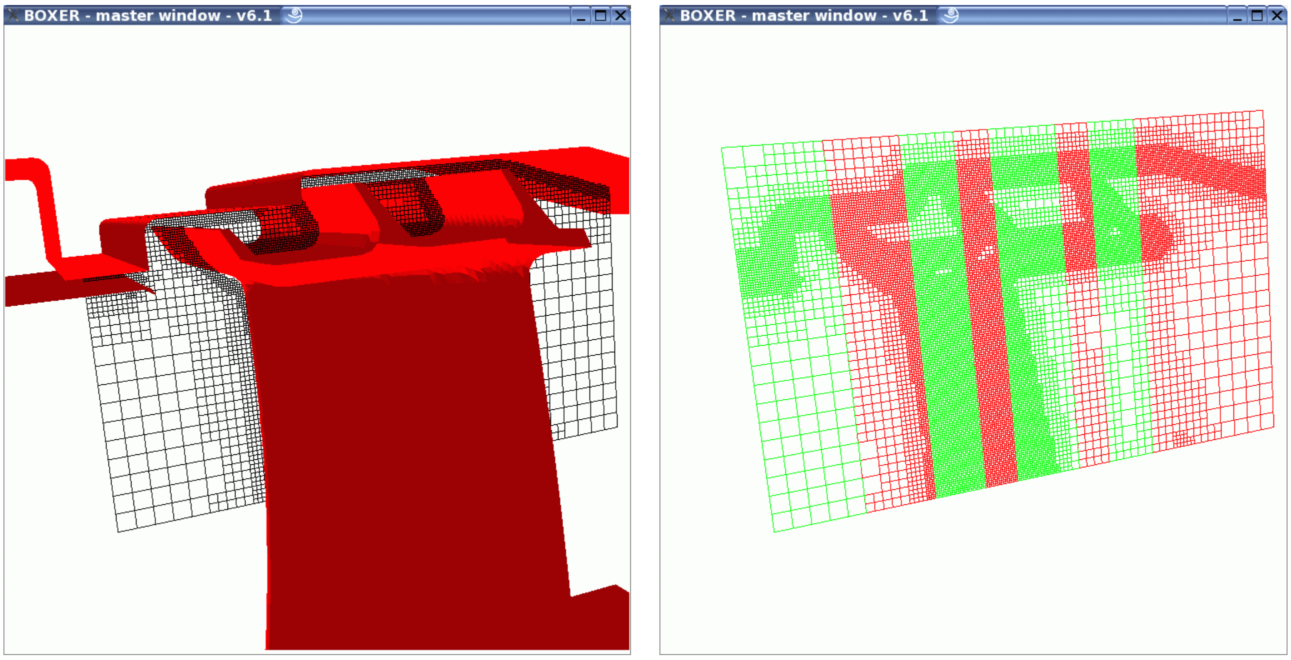


Figure 4 Capture of a generic industrial gas turbine geometry on a cluster of 8 PC's (the plot on the right hand side is coloured by processor number).

Figure 4 shows cuts through the turbine mesh: on the left hand side are the "flow cells"; on the right is the total mesh coloured to show the contributions of each of the 8 processors. The domain is partitioned by considering the X-wise distribution of defining boundary cells – hence the differing axial extents of each partition's mesh.

This static domain decomposition is simple to implement but not necessarily the most efficient since it merely shares the work roughly equally between processors rather than minimising communicate relative to compute loading. Future work will certainly focus on better partitioning and also dynamic as well as static load balancing.

B. New York – 5.36M cells

The next example was chosen to illustrate some of the opportunities that scalable, parallel mesh generation will open up for geometries of class city.

Increasingly, in areas of strong current interest like pollution dispersion, there is a need to link local building/street "micro" scale simulations based on CFD to large, regional "macro" scale simulations based on modelling. Hence, simulations combining elements of both CFD and modelling for the "meso" scales associated with entire cities are within sight.

The example chosen here was New York, available in STL or VRML formats readily on the internet (see for example: www.3dCADbrowser.com). Figure 5 shows two mesh cuts through a modest 5.36M cell mesh – this places only about 4-8 mesh spacings per building width but is sufficient to provide a realistic spatial distribution of blockage down at building/street scale. This is just too large to run well on a cluster of 4 PC's (the partitioning is not totally equal and one of the four PC's needs to run virtual) but runs very nicely on 8 and 16 PC's. The mesh shown in Figure 5 was generated on 16 PC's in only 52 seconds and the right hand plot is coloured to display the individual partitions.

The ability to generate a mesh so quickly from the tessellated input (input could also be taken from Digital Elevation Mapping datasets) means that it would be practical to start up a simulation in response to a particular pollution event for any city pre-stored in a library.

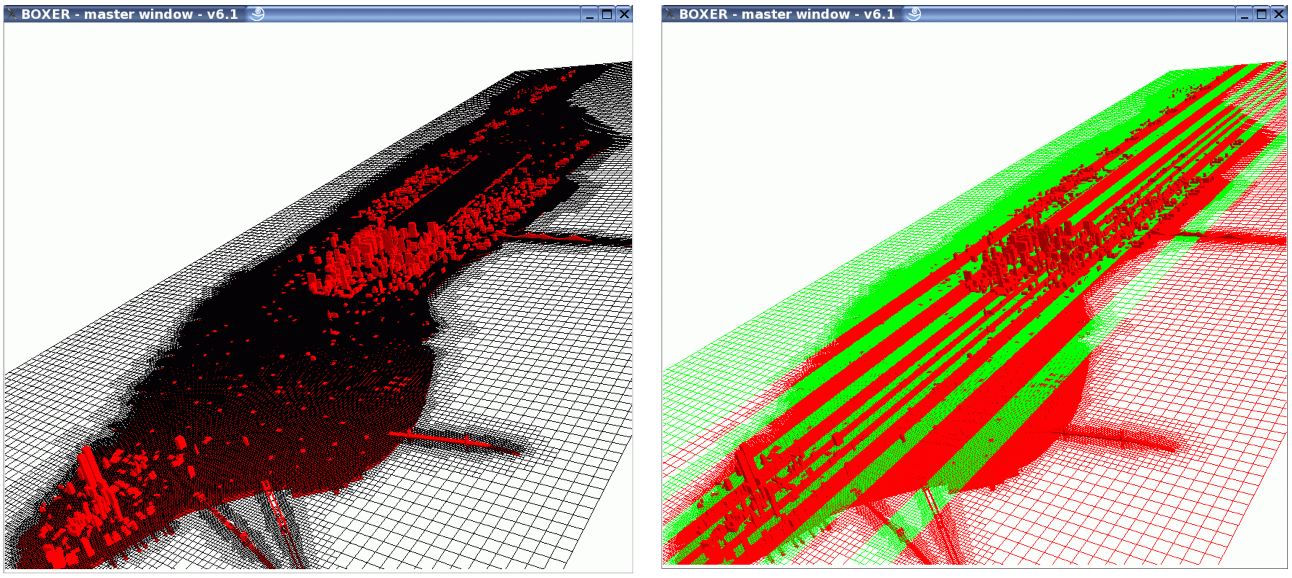


Figure 5 Two views of New York captured from a public domain VRML file; the plot on the right is coloured by processor number to illustrate the 16 partitions in use.

C. F18 aircraft -19.9M cells

The last example, illustrated in Figure 6, is a 19.9M cell mesh generated for a F18 aircraft imported in STL format. This mesh is too big for anything other than the complete cluster of 16 PC's – and is generated in only 64 seconds – neatly demonstrating the objective of pursuing fast, parallel meshing.

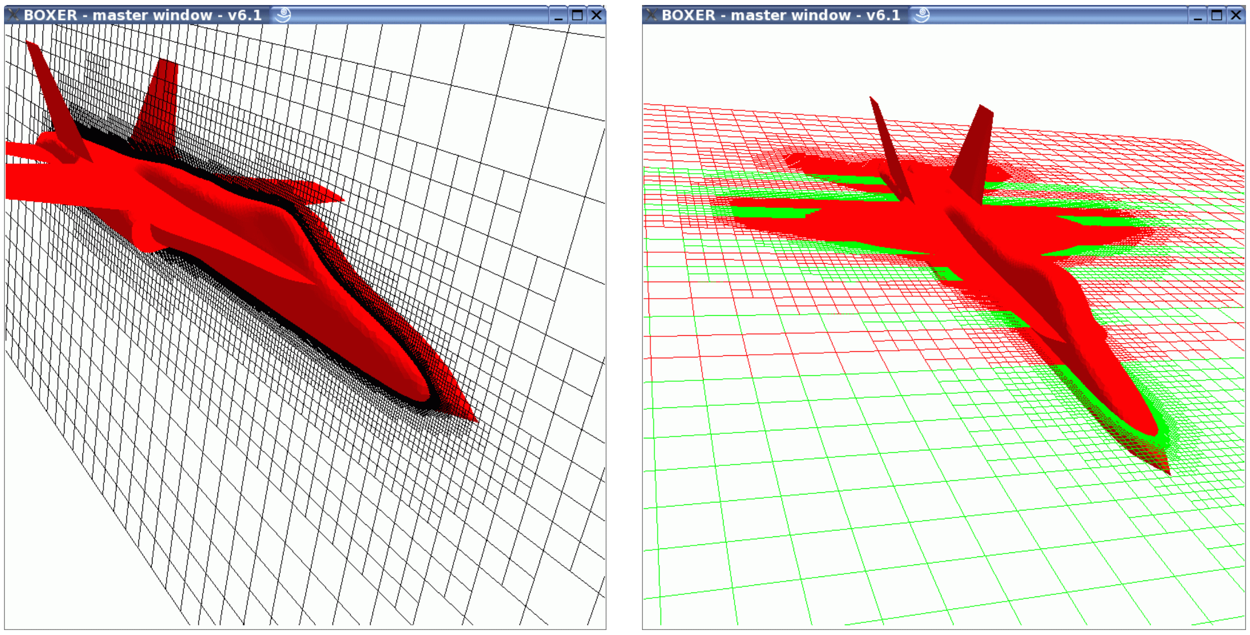


Figure 6 Views of a 19.9M cell mesh generated in 64 seconds on 16 PC's; the plot on the right is coloured by processor number.

IV. Concluding Remarks; Future Plans

This paper has attempted to show that it may be possible to construct a practical simulation system which is inherently parallel and scalable by integrating a solid modeling kernel, mesh generator, flow solver & post-processor within a single piece of software. The serial bottlenecks which represent an increasing restriction to more orthodox CFD Processes as geometries and problem sizes get bigger and more complex can be overcome by a radical rethink. Future work will try to exploit this new paradigm for rapid prototyping design optimization.

V. References

Adalsteinsson D & Sethian JA "A level set approach to a unified model for etching, deposition & lithography II: three dimensional simulations" J.Comput.Phys, 122, 348-366, 1995

Aftosmis MJ, Berger MJ & Melton JE, "Robust and efficient Cartesian mesh generation for component-based geometry" AIAA J., 36, 6, 952-, 1998

Baerentzen A "Volume sculpting: intuitive, interactive 3D shape modelling" IMM, May 2001

Bloor MIG & Wilson MJ "Using partial differential equations to generate free-form surfaces" CAD vol.22, no.4, pp.202-212, 1990

Bremer P-T, Porumbescu SD, Kuester F, Hamann B, Joy KI & Ma K-L "Virtual clay modelling using adaptive distance fields" ,xxx, 2001

Bussoletti JE, Johnson FT, Bieterman MB, Hilmes CL, Melvin RG, Young DP & Drela M "TRANAIR: solution-adaptive CFD modelling for complex 3D configurations" AIAA Paper 1995-xxxx, 1985

Dawes WN, Dhanasekaran PC, Demargne AAJ, Kellar WP & Savill AM, "Reducing bottlenecks in the CAD-to-mesh-to-solution cycle time to allow CFD to participate in design" ASME Journal of Turbomachinery, 2002

Dawes WN, Kellar WP, Harvey SA, Dhanasekaran PC, Savill AM & Cant RS "Managing the geometry is limiting the ability of CFD to manage the flow" AIAA Paper 2003-3732, 33rd AIAA Fluid Dynamics Conference, 23-26 June 2003, Hilton Walt Disney World, Orlando

Dawes WN "Building Blocks Towards VR-Based Flow Sculpting" 43rd AIAA Aerospace Sciences Meeting & Exhibit, 10-13 January 2005, Reno, NV, AIAA-2005-1156

Galyean TA & Hughes JF "Sculpting: an interactive volumetric modelling technique" ACM Trans., Computer Graphics, vol.25, no.4, pp 267-274, 1991

Glassner AS "Graphics Gems" Academic Press, 1990+

Haimes R "pV3…", xxx, 1997

Haimes R & Follen GL "Computational Analysis Programming Interface" Proc. 6th Int. Conf. On Numerical Grid Generation in Computational Field Simulations, Eds. Cross, Eiseman, Hauser, Soni & Thompson, July 1998.

Harvey SA, Dawes WN & Gallimore SJ "An Automatic Design Optimisation System for Axial Compressors Part I: Software Development" ASME Paper GT2003-38115

Harvey SA, Dawes WN & Bolger JJ "An Automatic Design Optimisation System for Axial Compressors Part II: Experimental Validation" ASME Paper GT2003-38650

Jones MW & Satherley R "Voxelisation: modelling for volume graphics" , xxx, 2000

Kellar WP, Savill AM & Dawes WN "Integrated CAD/CFD Visualisation of a Generic F1 Car Front Wheel Flowfield" Lecture Notes in Computer Science, Vol 1593, 1999

Kellar WP "Geometry modelling in CFD and design optimisation" PhD Dissertation, Cambridge 2003

Lamousin HJ & Waggenspack WN "NURBS-based free-form deformation" IEEE Computer Graphics & Applications Vol.14, No.6, 1994

Osher S & Sethian JA "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations" J.Comput.Phys, 79, 12-, 1988

Perng K-L, Wang W-T, Flanagan M & Ouhyoung M "A real-time 3D virtual sculpting tool based on modified marching cubes", SIGGRAPH, 2001

Samareh, JA "Status & Future of Geometry Modelling and Grid Generation of Design and Optimisation" J.of Aircraft, Vol 36, no1, Feb 1999

Samareh JA "Survey of shape parameterisation techniques for high-fidelity multi-disciplinary shape optimisation" AIAA Journal Vol.39, No.5, 2001

Sederberg TW & Parry SR "Free-form deformation of solid geometric models" Computer Graphics Vol.20, No.4, 1986

Sussman M, Smereka P & Osher S "A level set approach for computing solutions to incompressible two-phase flow" J.Comput.Phys, 114, pp 146-159, 1994

Viecelli JA "A computing method for incompressible flows bounded by moving walls" J.Comput.Phys, 8, 119-, 1971

Yerry MA & Sheppard MS "Automatic three-dimensional mesh generation by the modified octree technique" Int.J.for Num.Methods in Engineering, vol.20, 1965-1990,1983