8th. World Congress on Computational Mechanics (WCCM8)
5th European Congress on Computational Methods in Applied Sciences and Engineeering (ECCOMAS 2008)
June 30 –July 5, 2008
Venice, Italy

Harnessing the power of multi-core architectures in CFD by coupling multi-threading programming, cache misses reduction techniques with multi-scale adaptivity

*** Adrien Loseille[1], Frédéric Alauzet[1], Paul-Louis George[1], Loic Maréchal[1] and Éric Saltel[1]**

[1] INRIA, Gamma team
Domaine de Voluceau - Rocquencourt
78153, Le Chesnay, France
Adrien.Loseille@inria.fr
http://www-c.inria.fr/gamma/gallery/

Key Words: *Multi-threading Programming, Cache Misses Reduction, Multi-scale Anisotropic Mesh Adaptation, Large Steady and Unsteady Inviscid Flow Simulations.*

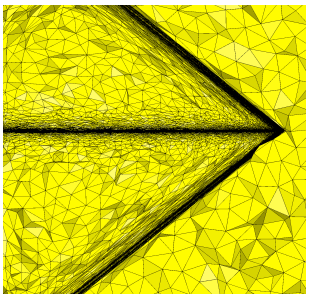
ABSTRACT

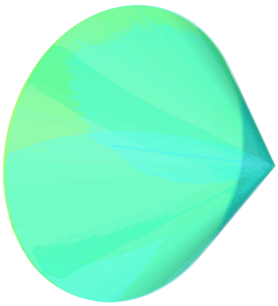
The wish of developing efficient parallel codes is generally driven by several requirements and practical considerations: the problem at hand that need to be solved, the required level of accuracy and the available computational power. In our case, we are interested in aerodynamic design which requires to perform a large number of CFD computations on complex geometries as accurately as possible within a limited amount of time, generally one week. To reduce the problem complexity, several solutions may be considered. A classical one is massive parallelization. However, we propose another strategy that is complementary to parallelization. At first, we utilize anisotropic mesh adaptation techniques to reduce the mesh complexity by exploiting in particular the natural anisotropy of physical phenomena. With the emergence of cheaper and cheaper multi-core computers on the market, we propose to use parallel algorithm on share memory machine based on multi-threading programming. Our strategy consists in coupling multi-scale anisotropic mesh adaptation with multi-core parallelization. One of the main assets of this strategy resides in a slighter impact on the solver's code implementation and the numerical algorithms.

The scope of this talk is the study of some implementation concepts designed for multi-core architectures. In addition to classical issues, as cache misses reduction and data reordering, our approach needs to be efficient on strongly anisotropic meshes. Contrary to structured or isotropic adapted meshes, no spatial compactness of the mesh vertices distribution with respect to the elements is assumed. In other words, anisotropic elements often connect points that are far from each other. We propose to study and compare several ways to parallelize the more CPU demanding tasks that are involved in our adaptive loop: (i) the Euler flow solver based on the mixed-element-volume method and (ii) the unstructured mesh optimization. In each case, the classical gather/scatter techniques that are usually developed on

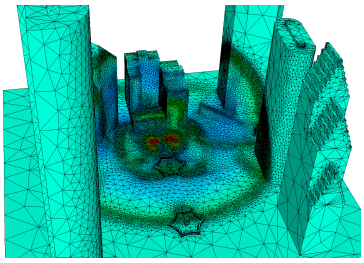
distributed memory architectures have been implemented and compared to our "overhead quasi-free" implementation. We emphasize that the gather/scatter methodology does not fit multi-core architectures. This is mainly due to the impossibility of accurately measuring the CPU time of each thread. Consequently, no reliable and guaranteed load balancing procedure can be deduced. To this end, we propose a dynamic load balancing processed on the fly. It is based on a dynamic memory block splitting strategy on which a memory access hierarchy and compatibility is computed. The overhead time is thus equal to the number of remaining blocks that are incompatible. Consequently, the efficiency of such approach is directly connected to the reordering procedure used. To give a quantitive analysis on the efficiency of our approach, the impact of using sort algorithms and space filling curves is also proposed.

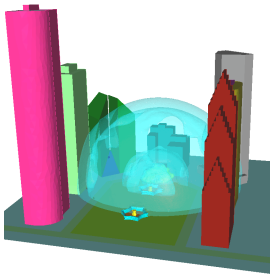
With a well chosen reordering strategy, this approach results in a very low memory overhead and a scalable algorithm. The overhead time is mainly due to synchronization as there is no more need to scatter and gather data. Moreover, minor modifications of the serial code are required to get the parallel version. Every optimization on the serial code have the same impact in parallel which makes code maintenance easier. The efficiency of this approach is demonstrated on 3D transient and steady simulations, see below. CPU time, overhead and scalability are studied on architectures having 2 and 4 dual-core. Moreover, a parallel computation of the flow around an 3D aircraft will be performed on a multi-core laptop during the talk to demonstrate the efficiency of the proposed strategy.





With cache optimization reordering time: 6s			
# proc.	CPU time (s)	Speed up	Gather Scatter time(s)
1	2783		
2	1446	1.9	18
4	751	3.7	18
8	464	6	28





Without cache optimization			
# proc.	CPU time (s)	Speed up	Gather Scatter time(s)
1	9279		
2	5200	1.78	19
4	3060	3	19
8	2294	4	29

Examples of steady and unsteady simulations with adapted anisotropic and isotropic meshes. Top, study of the sonic boom of a supersonic aircraft. Bottom, simulation of a blast propagation in a 3D town. Right, speed-up results obtained on an adapted mesh composed of $22,400,000$ tetrahedra, needed memory is 3 Giga bytes, overhead memory is 8% per thread.

REFERENCES

[1] G. Jin and J. Mellor-Crummey. "Using Space-filling Curves for Computation Reordering". *Proc. of the 6th Los Alamos Computer Science Institute*, NM, 2005.

[2] R. Lohner . "Renumbering Strategies for Unstructured-Grid Solvers Operating on Shared-Memory, Cache-Based Parallel Machines". *AIAA paper*, 1997-2045, 1997.

[3] A. Loseille, A. Dervieux, P. Frey and F. Alauzet. "Achievement of global second-order mesh convergence for discontinuous flows with adapted unstructured meshes". *AIAA Paper*, 2007-4186, 2007.