HPC for Large Scale CFD in Aircraft Design

Mattias Sillén

Saab Aerosystems, SE-581 88 Linköping, Sweden
Mattias.Sillen@saab.se

Abstract. In aircraft development the aerodynamic design is a long lead item. To reduce the development time advanced flow modeling methods must be used earlier than today. This is however limited by the current turn around time for the simulations. To reduce the turn around time the computer platform is very important. With the recent popularity of Linux–clusters it is now possible to design cost efficient systems for a specific application. A flow solver is investigated for parallel performance on various clusters. Hardware and software factors influencing the efficiency are analyzed and recommendations are made for cost efficiency and peak performance.

1 Introduction

The current challenges in the aerospace field are to offer products that are both better in performance and also faster and cheaper to produce. This forces aircraft designers towards risk minimization and reduction in cost and time to market. The possibility to influence life cycle cost is largest in the early design stages. Hence, the confidence in the design during the early phases must increase compared to today. This requires high fidelity flow simulations to enter the aerodynamic design process earlier than today and also the use of more advanced flow modeling methods throughout the complete design process. However, the turn around time for a simulation is a restricting factor when deciding the physical modeling level in each design phase. Parallel processing is commonly used to reduce the turn around time and with the introduction of PC-clusters in the mid-1990s cost-efficient supercomputing resources has become available to a wider community. Larger and application specific computer systems are now designed using cheap commodity components. The selection of node configuration and interconnecting network are factors influencing the parallel performance. Making the better choices will provide improved performance of the simulations and offering a chance to improve the modeling capability or reducing the turn around time. This paper discusses the impact of these choices on the application performance. Implementation and load balancing issues are also investigated.

2 Physics and Parallel Implementation

This study is based on results obtained with the unstructured Navier–Stokes solver Edge [1], developed at the Swedish Defence Research Establishment (FOI). The

solver has an edge based formulation that makes it possible to compute on any type of mesh: structured, unstructured (with tetrahedral, hexahedra, prism or pyramids) or hybrid. The flow equations are approximated on the computational mesh using a finite volume approach and the spatial discretization is done either with a cell centred scheme with added artificial dissipation or an upwind scheme, all second order accurate. A Runge–Kutta method is used to integrate the flow equations forward in time. The convergence to steady state is accelerated by an agglomeration multi grid method, where the solutions on a sequence of coarser grids are combined to improve the convergence rate.
Edge is parallelized by domain decomposition with MPI as the message passing system. In the parallel implementation the same processor operates on all multi grid levels of a partition. The load balancing is performed with the graph partitioner software Metis [2] on the finest grid level. Control volumes on coarser levels are assigned to the partition that contains the largest part of each individual control volume. This minimizes the communication between processors when changing grid level but may lead to load imbalance on coarser grid levels. An alternative is to perform load balancing on each grid level separately. This will increase the amount of communication when changing grid level but will guarantee a better load balance also on coarser grids levels. The communication pattern between the processors is predetermined at run–time following the domain decomposition. The transfer of data is executed by packing data from all boundary cells on a given processor that are to be sent to another processor into a buffer that is sent as a single message. This standard approach for the inter–processor communication has the effect of reducing latency overheads by creating fewer and larger messages.

3 Cluster Performance Evaluation

To evaluate the code performance on different clusters configurations a representative model from aerodynamic design is used. The present model was used for a study on the aerodynamic effects during a store release from the Gripen fighter; see Fig. 1 for the surface grid. The case is geometrically complex with detailed external stores placed underneath the wings. A total of 3 million nodes corresponding to approximately 18 million tetrahedral volume elements are needed for a full span model to adequately resolve the geometry and the inviscid flow features. A fully converged steady–state solution can be achieved in about 500 multi grid cycles. Often a large number of cases, typically 50-100, with different flow conditions and store locations are computed. The present case is at transonic conditions with sideslip, Fig. 1 also presents the pressure distribution on the upper side of the aircraft.
A fixed size problem is used as the focus is on industrial applications and the intention is to reproduce the situation in the design process. When the problem is parallelized over more processors two parts will influence the performance results more than the other. First, the computation to communication ratio will decrease as the partitioning introduces new internal boundaries between domains. Both the total amount of data communicated as well as the number of messages increase. The communication pattern becomes more fragmented and the mean message size decreases. Secondly,

when more processors are added the total amount of fast cache memory also increase. This means that a larger part of the total problem will reside in the cache with a subsequent performance gain. This is called cache effect and can result in a super linear speedup, i.e. higher speedup numbers than number of processors.

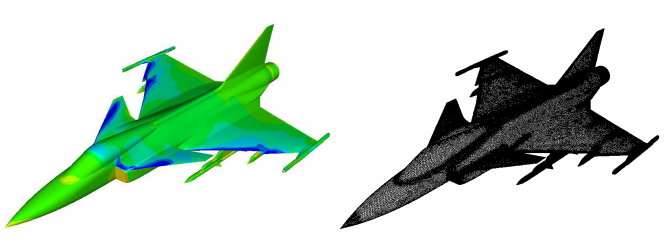

Fig. 1. Transonic flow around the Gripen fighter, pressure distribution and surface mesh.

Three types of cluster configurations are evaluated. One with a commodity network (Gigabit Ethernet) and two with high performance networks (SCI and Infiniband). They differ both in latency, 30 µs compared to 3-7 µs, and bandwidth, 70 MB/s compared to 300-600 MB/s.
On a SCI cluster with dual nodes an aggregated computational performance of 34 GFlops is reached on 256 processors. Using only one processor per node it delivers 21 GFlops on 128 processors, see Fig. 2.

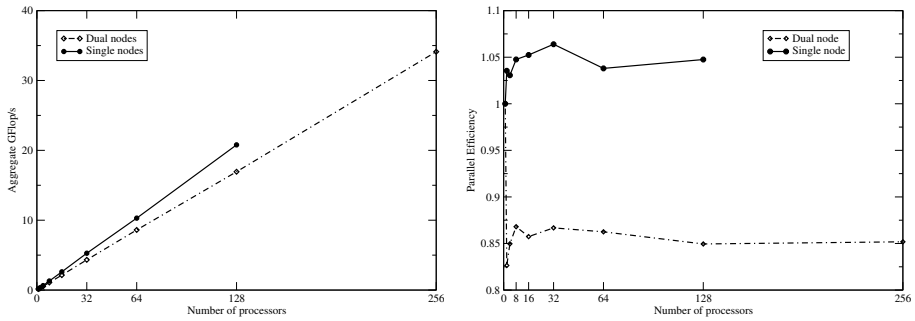

Fig. 2. Computational performance and parallel efficiency on dual and single node SCI cluster.

A notable observation is that when using nodes with dual processors, which may be attractive from the point of view of cost–efficiency when using an expensive network or from compactness aspects, the performance is reduced 20 % when two processors have to share on a common node memory. The memory bandwidth is in this case not up to the demands of the memory intensive application. The application demonstrates a nearly linear parallel speed-up. This is also seen in the right graph in Fig. 2 where the parallel efficiency stabilizes on 1.05 for the single processor node and 0.85 for the dual node. From this we conclude that the network capacity is sufficient at least up to 256 processors.

Comparing performance from clusters with Gigabit Ethernet, SCI and Infiniband in Fig. 3 demonstrates that the Gigabit Ethernet cluster rapidly looses in efficiency already after 8 processors. This is not the case for the other networks and the main reason is the lower latency. Analyzing the communication behavior of the code reveals that the communication pattern quickly gets latency bound. Already at 8 processors the mean message transfer time is affected by latency. The much higher efficiency values for the Infiniband configuration is caused by cache effects due the fixed size problem. The nodes in this configuration are equipped with much larger 2nd level cache than the other configurations.

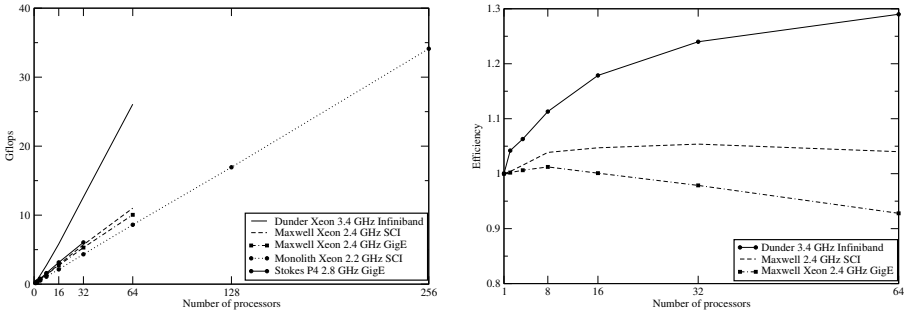

Fig. 3. Computational performance and parallel efficiency on clusters with differing network.

From this investigation it can also be concluded that good load balancing can be achieved with the software Metis. Up to 32 processors it is sufficient to balance only the number of points in the partitions but above it also important to minimize and load balance the communication load (boundary points) between the processors.

4 Conclusion

Different cluster configurations are evaluated for performance using an unstructured flow solver in an industrial environment. High performance networks delivers excellent speed-up for hundreds of processors. The low-cost alternative Gigabit Ethernet performs well up to 100 processors and with single processor nodes it is the most cost-efficient alternative. Dual processors loose performance due to memory bus saturation but are still recommended together with high-end networks due to cost reasons.

References

1 Eliasson, P.: EDGE, a Navier-Stokes solver for unstructured grids. Proceedings of the Finite Volumes for Complex Applications III, ISBN 1 9039 9634 1. (2003) 527-534
2 Karypis G. and Kumar V.: Analysis of Multilevel Graph Partitioning. Technical Report 95-037. University of Minnesota. (1995)