Heterogeneous Parallelism of Aero-Acoustic Applications Using PACX-MPI

Harald KLIMACH¹, Sabine P. ROLLER¹, and Claus-Dieter MUNZ²

¹High Performance Computing Center Stuttgart (HLRS), University of Stuttgart, Nobelson 19, 70550 Stuttgart, Germany
²Institute for Aerodynamics and Gasdynamics (IAG), University of Stuttgart, Pfaffenwaldring 21, 70569 Stuttgart, Germany

Received February 29, 2008; final version accepted December 1, 2008

This paper outlines the distribution of an aeroacoustic application in a heterogeneous super computing environment. PACX-MPI, which is used for this distribution allows the coupling of different architectures without leaving the MPI context in the application itself. This makes the usage of a heterogeneous infrastructure very convenient from the applications point of view.

Integrated simulation of fluid flow with its aeroacoustic is a typical multi-scale task with different numerical requirements in the involved parts. The different requirements can be spatially separated as the noise generating object is generally rather small when compared to the area which is to be computed for the sound propagation. A natural division into a domain of noise generation and a domain of noise propagation arises and the two different computational domains are only loosely coupled with numerical requirements very distinct from each other. We demonstrate how the parallel simulation of a 3D aeroacoustic testcase can benefit of the heterogeneity in the infrastructure by mapping the heterogeneous computational domain on the appropriate architectures.

KEYWORDS: Aeroacoustics, Coupled Applications, Parallel Applications, Computer Architectures

1. Coupled Aeroacoustic Application

To simulate the generation of sound waves and their propagation efficiently, a coupling scheme is needed to allow different computations in each part. Direct simulation of the multiscale aeroacoustic problem involves the solution of the flow field including as many physical flow effects as possible and the solution of the sound wave propagation with this waves as dominating phenomena of interest. Thus the flow field should be computed using the full Navier-Stokes equations, whereas it is sufficient to solve the linearized Euler equations for the wave propagation. Because in the usual setup there is only a rather small obstacle in the scale of meters generating the noise in the flow and the wave propagation is to be simulated in a much larger area in the scale of kilometers it is possible to spatially divide the both domains with different numerical requirements. Only a relatively small area surrounding the obstacle has to be computed with the full Navier-Stokes equations and a high spatial resolution. The computation of the level of detail, necessary for the flow field effects, throughout the complete aeroacoustic domain would be a waste of computational resources, so allowing different computations by coupling heterogeneous domains together is essential.

Our application is mainly developed for high speed flows and thus provides a coupling mechanism for explicit time marching solvers only. There are two kinds of solvers involved. One for structured meshes and one for unstructured meshes. Both can use Finite Differences (FD), Finite Volume (FV) or Discontinuous Galerkin (DG), described for example by Hesthaven and Warburton (2007) as spatial discretization method with high order. Time integration is done using the “Arbitrary high order using derivatives” (ADER) scheme.

The used coupling preserves the high order of the numerical scheme across domain interfaces. It is realized using ghost cells to exchange information between the different computational domains. Construction of the ghost cells can be reduced to values at discrete points for all three used spatial discretization methods. By the reduction to discrete points the coupling becomes independent of the discretization, as each domain only needs to interpolate its own solution onto an exchanged set of coordinates to provide all necessary data to its neighbors. In each domain the ghost cells can then be reconstructed by just using the exchanged values at the appropriate points. The discrete points are equally easy to handle in 3D as in 2D and provide a path to data encapsulating for the parallel computation. The coupling ghost cells of one domain interacting with two neighbors are depicted in Fig. 1. For the ghost cell overlapping both neighbors, the constructing discrete points are indicated. Not shown are the ghost cells of the other two domains.

For parallel computation of the domains, a list of all the points, which can be found in a neighbor, has to be created for each neighbor and then exchanged over MPI Point-to-Point communication.
To really take advantage of different discretizations in the domains it is also necessary to allow different time steps in the domains. The used coupling scheme is capable of this by using a sub-cycling mechanism, where time steps in domains are allowed to be multiples of the smallest timestep. Data has only to be exchanged when neighbors reach a common time level. During the larger time step the domain with the smaller time step has to approximate the value of its ghost cells at each time level within the longer time step of the neighbor. A more detailed description of the complete coupling scheme is given by Utzmann et al. (2006).

In the view of parallel computations, this kind of coupling results in few communication steps with only local neighbors involved, and only the discrete point values to be exchanged. Hence, the coupling between different domains can be realized even with quite low network requirements. This makes it well suited for inter-cluster communication in the heterogeneous network.

2. PACX-MPI

The PACX-MPI library was developed to enable the usage of a heterogeneous set of supercomputers without leaving the MPI context. A more detailed description can be found in Beisel et al. (1997). From the application point of view there is no difference to be seen when running in PACX-MPI instead of a local MPI. All processors across the heterogeneous setup are collected within the global communicator, regardless of the actual physical position of the processor. All these processes are then equally available to the application just as in an ordinary MPI execution.
PACX-MPI uses on each machine the “native” MPI library underneath which is generally optimized for the corresponding cluster. The inter-cluster communication is done by two daemons on each side. Any communication which happens between processors on different clusters is detected by the library and handed over to those two processes. In Fig. 3 the resulting process layout is shown. The advantage of this layout is that PACX-MPI can account for different network protocols within each machine and between the clusters. This is important, as the clusters may provide fast Infiniband interconnects while the link between the different machines may just be a slower ethernet interconnection.

The two daemon processes for inter-cluster communication are responsible for the local distribution of messages, apply data conversions if necessary and can compress data, that is to send over the slower network connection between clusters. The setup of fast clusters connected by not as fast communication lines is especially well suited for loosely coupled applications as the aeroacoustic example we are looking at here, but it even can serve strongly coupled applications quite well as found by Boenisch and Ruehle (1998).

3. Operating the Application in PACX-MPI

As presented above, the aeroacoustic application offers an appropriate method to couple heterogeneous domains with relatively little communication. Each involved domain may have different numerical requirements. PACX-MPI on the other hand offers an elegant way to run an application distributed across several super-computing clusters with architectures offering different features.

Our goal is to turn the heterogeneity of both the simulated flow and the super-computing environment into an advantage by fitting the requirements of each part in the simulation to the properties of the used hardware. In order to allow this we added a property to the running processes by which it can be decided in the application on which architecture it is actually running. This is defined on the commandline for the application on each cluster and can then be used during the distribution of the domains onto the available processes to map each domain to the best suited architecture. Especially we can run the structured domains on a vector machine NEC SX-8, available at the High Performance Computing Center Stuttgart (HLRS), and the unstructured ones on a cluster with scalar architecture. This division is done once at startup time, therefore the proper ratio of computation loads for each cluster has to be known a priori. Generally an estimation run is necessary to figure out the resource distribution for a balanced run.

3.1 Communication Layout

As already discussed, the aeroacoustic application actually consists of several parts acting together. There are the solvers acting on each domain and the coupling between those domains. Each solver in the coupled domain can be spread across several processes for computation. The parallel computation within the domain is done by partitioning the domain using Metis for the unstructured meshes or a simple cartesian partitioning for the structured meshes. The parallel coupling of the partitioned domains leads to an inherent hierarchy of communication levels. We call each partition of a domain a section. Each section is uniquely defined by the domain it resides in and the global rank of the process it is computed by. All communication in the application happens between the sections as fundamental units.

Neighboring sections communicate by MPI point to point messages. For internal communication inside each domain, which is necessary at every timestep, a communicator for all processors doing computations in that domains is defined. Data of the discrete points for the ghost cells responsible for the coupling between different domains is sent using the global communicator.

This communication layout maps quite well to the existing network setup. Generally there is only a weak interconnection between different clusters but strong connections between processors within one cluster. Thus good performance may be gained, when the domain boundaries of the application coincide with the cluster boundaries. As within each domain more communication is necessary at every time step a single domain should be ensured not to be spread across different clusters.
3.2 Results of the Heterogeneous 3D Simulation

The aeroacoustic 3D problem we are using as an example here is the scattering of sound waves emitted from a single point at a perfectly reflecting sphere. This was proposed by Morris (1997) for the Second CAA Workshop on Benchmark Problems. The exact solution of this problem is known and so the results of the distributed run can be validated.

There are two domains involved in this simulation. An unstructured mesh around the sphere approximates the geometry and solves the flow near the sphere using the discontinuous Galerkin scheme. This domain is embedded into a structured mesh simulating the far field using the finite differences scheme. In both domains an 8th order spatial discretization is used. The sphere has a radius of one meter and the whole simulated volume has an extent of $102.2 \times 57 \times 57$ meters. The structured mesh has about 42 million cells and the unstructured mesh is built with 9874 cells.

The large structured FD domain can be computed very efficiently on a vector machine, whereas the unstructured DG domain can not take advantage of the vector architecture. Thus the simulation is spread on a heterogeneous setup with a NEC-SX8 part on the one side and an NEC-TX7 on the other. A single NEC-SX8 vector CPU has a theoretical peak performance of 16 GFLOP/s, 8 processors are grouped together into a shared memory node, where each processor has access to the 128 GB of memory with a bandwidth of 64 GB/s. The nodes are connected by an internode crossbar switch (IXS) with a bandwidth of 16 GB/s. The vector CPU uses registers with a length of 256 numbers to perform a single operation on multiple data. As a frontend to the vector system serves a NEC-TX7, which is a shared memory system with 32 Itanium II processors operating at 1.5 GHz clock cycling speed and a theoretical peak performance of 6 GFLOP/s per CPU. It provides in total 512 GB of main memory, which each CPU can access with a bandwidth of 3.2 GB/s.

On the NEC SX-8 the main time consuming routine in the structured FD part can be operated with a performance of nearly 14 GFlop/s which equals to 87.5% of the peak performance. In contrast the unstructured mesh using DG spends most of its time in routines with less than 100 MFlop/s on that architecture.

To get representative results for the coupling behavior, considering only the different computation hardware, a single CPU of each architecture is used, so no communication within a cluster is necessary. The comparison of the involved architectures with respect to the different simulated parts is shown in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Linux Cluster</th>
<th>NEC SX-8</th>
<th>coupled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed time</td>
<td>27925 s</td>
<td>10966 s</td>
<td>3207 s</td>
</tr>
<tr>
<td>Unstructured</td>
<td>2994 s</td>
<td>7746 s</td>
<td>3019 s</td>
</tr>
<tr>
<td>Structured</td>
<td>23887 s</td>
<td>2871 s</td>
<td>2869 s</td>
</tr>
<tr>
<td>Coupling</td>
<td>1012 s</td>
<td>321 s</td>
<td>554 s</td>
</tr>
<tr>
<td>Waiting in MPI</td>
<td>0 s</td>
<td>0 s</td>
<td>164 s</td>
</tr>
</tbody>
</table>

The computation to simulate 2 seconds on a single Itanium II (IA64) processor elapses in total 27925 seconds of real time. On a single NEC SX8 processor the same simulation runs only 10966 seconds. However as shown in Table 1 the inefficient computation of the domain with the unstructured mesh takes more than twice as long as on the cheaper Itanium processor. To put this domain on the vector architecture would be a complete waste of resources and so it is a big advantage to put this part on a cluster of scalar processors. Even so some idling times are introduced by the not perfectly balanced computation the elapsed time of the coupled computation is better than what could be gained using two processors of only a single architecture.

This example nicely shows how the heterogeneous simulation can profit from the heterogeneous network of supercomputing clusters. For real simulations with a much more realistic far field in the range of kilometers and a much better resolved flow field around the sphere the need for computational resources rapidly increases and may easily surpass the power of a single super computing cluster. So in order to get an acceptable time to solution for those huge problems several clusters may have to be used and in order to avoid the waste of precious resources, it is required to run all computations on the best suited available architecture.

The parallel computation of the structured part does currently not work as expected in combination with the distribution of the domains. The problem is being worked on, but for now we don’t have results for scaled problems with more than one CPU in each domain. However with these promising results we are looking forward to use several nodes of the NEC-SX8 in combination with a distributed memory cluster as soon as the implementation issues are resolved.

3.3 Conclusion

In this work we built the basis for heterogeneous computations with architectural features of the hardware matching...
the numerical requirements of distinct domains involved in the direct aeroacoustic simulation. This was achieved by incorporating PACX-MPI after parallelization of the natural decomposition in the coupling application.

The remaining obstacle is the load balancing between the different architectures, as the load balancing has to be known before the jobs on the corresponding clusters are started. To overcome this it would have to be possible to shift processes from one cluster to the other one during the runtime. As this feature is currently not available an automated resubmitting of changed application jobs with reconfigurations may be a possible solution to improve load balancing over the run time of the application.

Further a convenient coallocation facility is needed to distribute a simulation across different clusters a to get concurrent timeslots in the queueing systems of the involved clusters. There is currently work going on in the Distributed European Infrastructure for Supercomputing Applications (DEISA) to provide this feature in combination with the fast interconnect between the different European supercomputing sites. We hope to exploit these facilities in combination with PACX-MPI as soon as they fully become available.

We are planning larger 3D simulations including more physical phenomena. Especially a simulation of the Von Karman vortex street behind a sphere would be an interesting validation example. Included in such a simulation would be the whole range of effects from the boundary layer to the aero acoustic wave propagation. This of course implies large scale parallelization of each simulation part.

REFERENCES