Control of turbulent boundary-layer flow using slot actuators

Ralf Messing, Ulrich Rist[1], and Fredrik Svensson[2]

[1] Institut für Aerodynamik und Gasdynamik (IAG), Pfaffenwaldring 21, 70550 Stuttgart, Germany [last name]@iag.uni-stuttgart.de
[2] NEC High Performance Computing Europe GmbH, Heßbrühlstraße 21B, 70565 Stuttgart, Germany fsvensson@hpce.nec.com

This paper describes the advances made by the collaboration between the Institut für Aerodynamik und Gasdynamik and the Teraflop Workbench. The target was to enable new research on the large SX-8 system at HLRS.

1 Introduction

For the case of flows over solid surfaces, the separation of the boundary layer causes large energy losses which in turn strongly affect the aerodynamic loads in terms of lift loss and drag increase. Therefore, there is a strong need to delay or even eliminate the occurrence of flow separation. Regarding commercial aircrafts the delay or elimination of separation of the typically turbulent boundary layer on the wing would permit higher angles of attack during landing and take-off. Using appropriate means for separation control one could even think of a high-lift system without slat (slatless wing) leading to devices with less maintenance effort and noise production.

In order to manipulate and control separated turbulent boundary layers jet actuators have been proposed injecting fluid into the boundary-layer flow by continuous or pulsed blowing. The influence of geometry and orientation of the orifices as well as the direction of the fluid jets (parallel, inclined or normal to the wall) is actually under examination in research studies. Basically, the application of these jet actuators aims at enhanced mixing rates in the boundary layer increasing momentum in the vicinity of the wall.

This paper presents a comparison of experiments conducted at the Institut für Strömungsmechanik at the Technical University of Braunschweig and direct numerical simulations done at the Institut für Aerodynamik und Gasdynamik.

1.1 Numerical Method

Details of the numerical method have been reported in various publications [Bon99, Mes04, Was02]. Therefore, the description of the numerical method can be restricted to the modifications that had to be carried out to simulate a slot actuator with steady blowing. Blowing is modeled by prescribing the steady wall-normal velocity at the wall:

$$v'(x,0,z) = v_c \cos^3\left(\frac{\pi r}{d}\right). \tag{1}$$

For spanwise slots with a slot width d_{SL} (extension in chordwise direction) and a slot length L_{SL} (extension in spanwise direction) follows :
For $z_{SL} \leq z \leq z_{SL} + L_{SL}$

$$d = d_{SL}\ ,\ r = \sqrt{(x - x_{SL})^2}\ ,\ r \leq \frac{d_{SL}}{2} \tag{2}$$

For $z < z_{SL}$

$$d = d_{SL}\ ,\ r = \sqrt{(x - x_{SL})^2 + (z - z_{SL})^2}\ ,\ r \leq \frac{d_{SL}}{2} \tag{3}$$

For $z > z_{SL} + L_{SL}$

$$d = d_{SL}\ ,\ r = \sqrt{(x - x_{SL})^2 + (z - (z_{SL} + L_{SL}))^2}\ ,\ r \leq \frac{d_{SL}}{2} \tag{4}$$

The slots have circular roundings at their lateral ends. The center of the rounding is located at $(x_{SL}, 0, z_{SL})$. In contrast to older versions of the numerical code the slots can now be rotated in the wall plane with respect to the main flow direction (skew angle β, see figure 1). Blowing is still perpendicular to the wall (pitch angle α, see figure 1).

2 Results

2.1 Undisturbed flow

Experimental data has been provided for a flate plate with zero-pressure gradient at a freestream velocity $U_\infty = 15m/s$ to gain some detailed insight into the effectiveness of skewed slot actuators. However, before comparing experimental and numerical results it has to be ensured that the turbulent bounday-layer flows without actuators match. To establish turbulent boundary-layer flow the numerical simulation is carried out according to the procedure in the experimental set-up. There, turbulence is triggered by an adhesive tape mounted downstream of the leading edge of the plate, and far enough upstream of the measurement station. A very similar approach is applied in the numerical

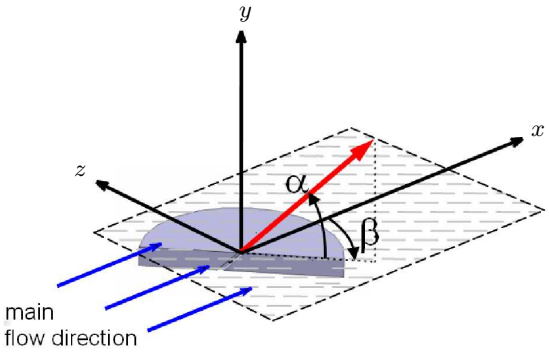

Fig. 1. Sketch of actuator for definition of pitch angle and skew angle

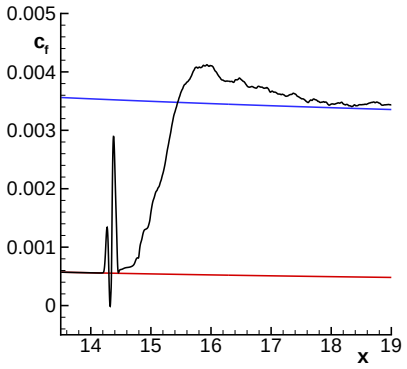

Fig. 2. Skin friction coefficient versus streamwise direction. Black line: simulation; Red line: laminar flow; Blue line: turbulent flow.

set-up. By harmonic suction and blowing in a disturbance strip unsteady disturbances are excited which lead to rapid breakdown of the initially laminar flow and rapidly provide a fully-developed turbulent boundary layer downstream of the disturbance strip. To illustrate this, the skin friction coefficient is plotted over the whole integration domain in figure 2. The disturbance strip is located at $x = 14.34$. Downstream the wall friction coefficient c_f strongly increases due to laminar-turbulent transition and rapidly approaches the values for fully-turbulent flow. Despite the penalty of additional computational time and memory requirements this approach has been prefered because it does not suffer from somehow unphysical initial boundary conditions for turbulent flow.

Finally, mean velocity profiles and rms-profiles of the streamwise velocity component at $Re_{\delta 1} = 1855$ are compared in figure 3 to mutually validate

experiment and numerical simulation. Agreement is quite satisfactory, even though some minor deviations are discernible in the rms-profiles near the wall.

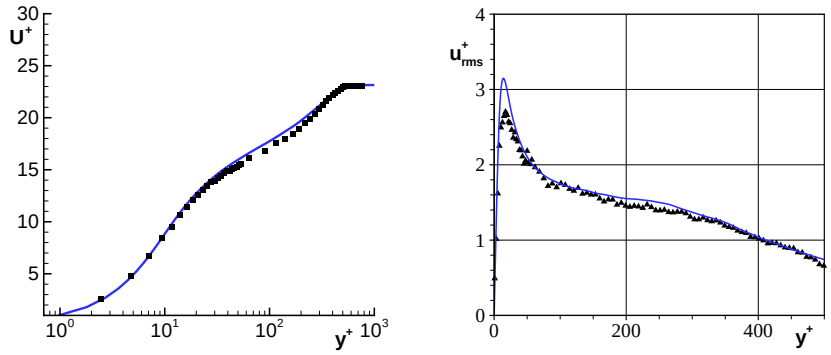

Fig. 3. Comparison of experiment (symbols) and simulation (lines) at $Re_{\delta 1} = 1855$; left: streamwise mean velocity profiles; right: streamwise rms-profiles

2.2 Disturbed flow

At the University of Braunschweig a measuring campaign was conducted to identify the most effective configuration to increase momentum near the wall. As mentioned a flate plate with zero-pressure gradient has been chosen to minimize expenses as extensive parametric studies were necessary. A slot with preferably steady blowing with a pitch angle $\alpha = 90°$ and a skew angle $\beta = 45°$ turned out to work best. The slot length is $L_{SL} = 10mm$, the slot width is $d_{SL} = 0.3mm$. The maximum blowing velocity is $v_{max} \approx 75m/s$, which is about five times the freestream velocity.

The small slot width in combination with high blowing velocities constitutes very challenging boundary conditions for the direct numerical simulations. In order to resolve all occuring flow scales a very fine grid, especially in wall-normal direction, must be used. The main parameters are set as close as possible to the experiment and are finally: $L_{SL} = 10mm$, $d_{SL} = 2mm$, $v_{max} = 40m/s$.

Although not all main parameters exactly match, qualitative agreement between experiment and numerical simulation is quite encouraging. To illustrate this, streamwise mean velocity contours and cross-flow velocity vectors in three successive cross sections downstream of the slot are plotted in figures 4 and 5. The strong blowing acts like a large obstacle and downstream of the slot actuator a vortex forms transporting high-momentum fluid to near-wall regions. This vortex is persistent and still increases near-wall momentum far

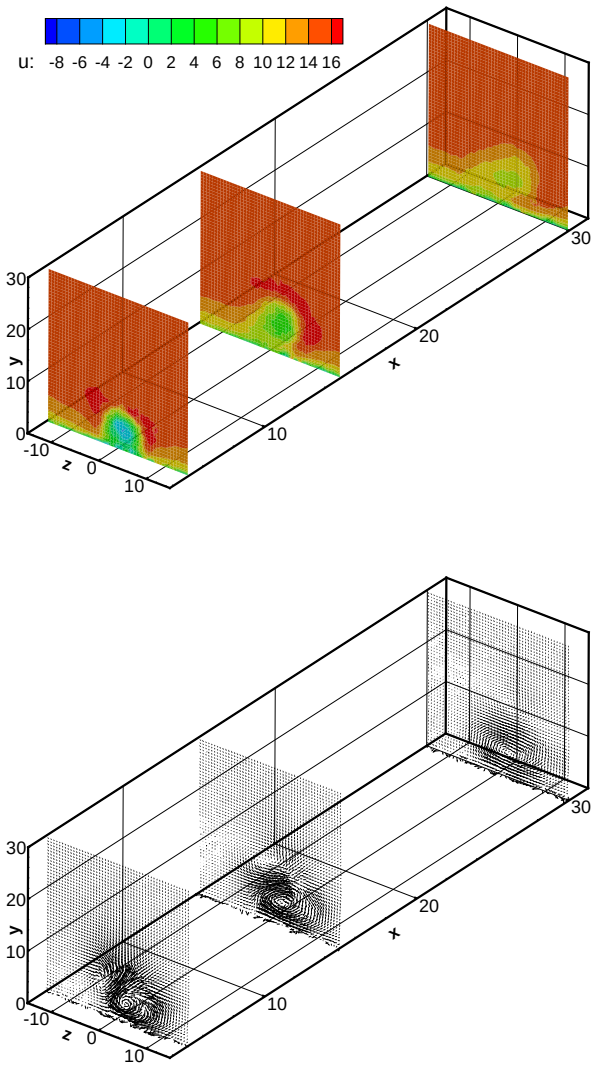

Fig. 4. Contour lines of mean streamwise velocity u (in m/s) (top) and v-w-vectors (bottom) at three planes across main flow direction. Data from experiments provided by University of Braunschweig [Sch06].

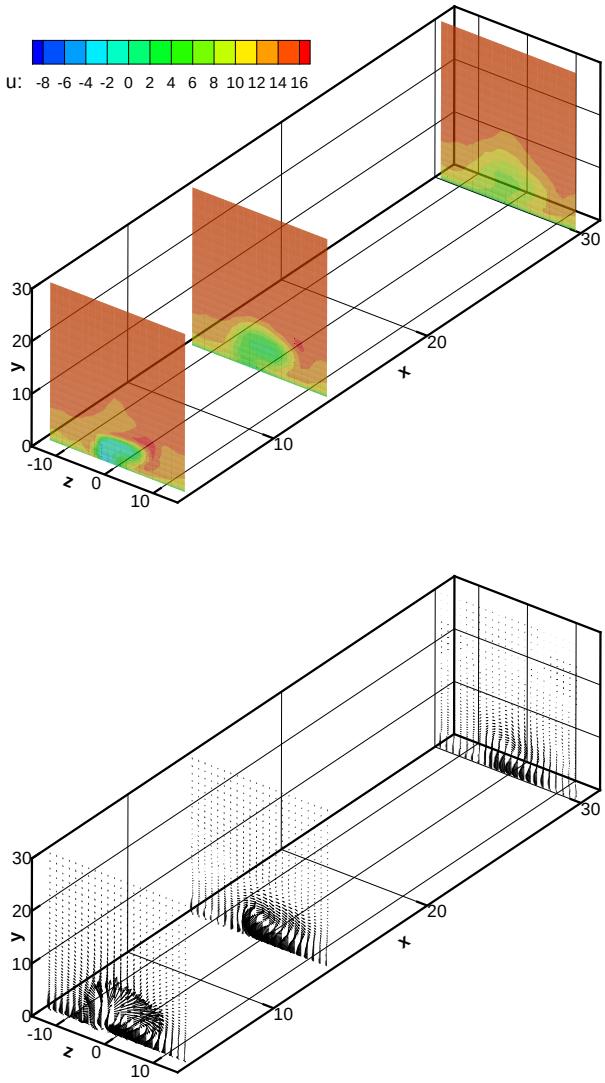

Fig. 5. Contour lines of mean streamwise velocity u (in m/s) (top) and v-w-vectors (bottom) at three planes across main flow direction. Data from unsteady direct numerical simulations.

downstream of the actuator. This mechanism is clearly observable in experimental data and qualitatively reproduced by the numerical simulation.

3 Conclusions

A comparison between experiment and unsteady direct numerical simulation of a slot actuator with steady blowing for turbulent boundary-layer separation control has been presented. During experimental studies it turned out that high blowing velocities are required to increase momentum near the wall, and therefore to hinder the flow to seperate from the flow surface in adverse-pressure gradient flows. The numerical effort to simulate such control devices is tremendous, at least with the actual formulation of our DNS solver. Improvements are planned or already under way, like the use of a strechted grid in spanwise direction or domain decomposition with a refined grid in the vicinity of the actuator.

4 TeraFlop Workbench Tunings

Within the TeraFlop Workbench project tunings were introduced to the code to enable new research.

4.1 Introduction

The application was reworked in several stages to provide better scalability and better single CPU performance. The improvements are to enable better utilization of a large machine. Traditionally the program has been run on one maybe two SX nodes. The high performance of the SX-6 machines in comparison to their contemporary competitors made the SX the platform of choice. As single CPU systems or even SMP machines have reached the limits of what is possible in performance the direction is to run constellations of fast machines. This introduces new problems as domain decomposition.

Scaling becomes very important and even if a problem is dividable most problems are not easily parallelized. As the machine is very large with 72 nodes, one wants to take advantage of the power of these nodes to calculate larger and more detailed cases. The application scales with the dimensions of the dataset, this means that distributing the code on less powerful processors, increases the need for a larger domain, which in turn increases the need for computing power. The SX-8 nodes in themselves already are very powerful and by simply scaling the problem from the old usage of one to two nodes, to ten nodes improves throughput and research abilites of the IAG.

4.2 Tunings

Sine and Cosine transforms

The application is relying on sine and cosinus transforms, since the Z dimension is represented in frequency space. These transforms were compiled from source. The sine transform uses sines only as a complete set of functions in the interval from 0 to 2π, and, the cosine transform uses cosines only. By contrast, the normal Fast Fourier Transform (FFT) uses both sines and cosines, but only half as many of each. Sine and cosine transforms are not a part of the highly optimized mathematical libraires, however FFTs are. By combining sine and cosine transform data, it is possible to use the FFT to do the transform.

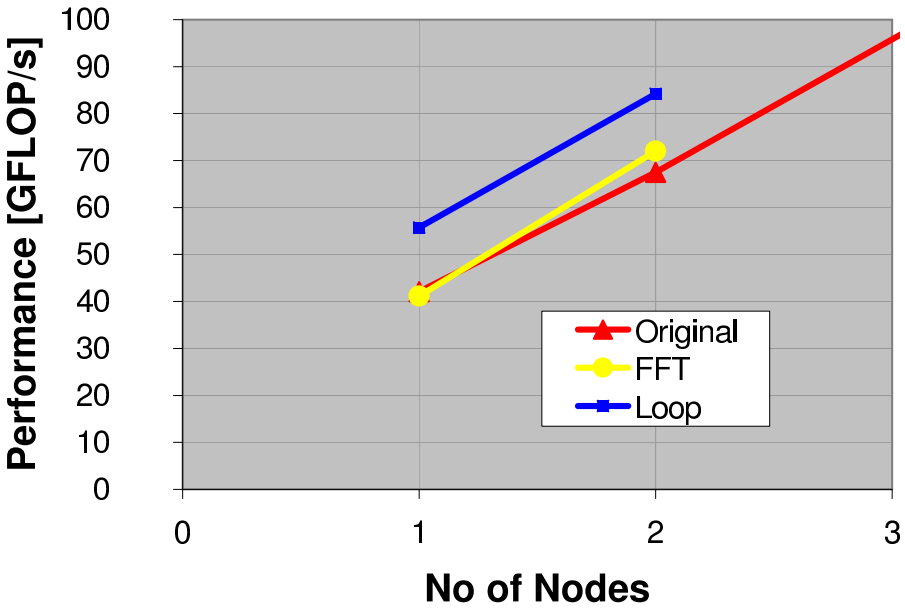

Fig. 6. Improvment by switching to FFT is not so visible in the performance plot as the number of operations went down with the execution time. Single CPU loop improvements also give the performance a boost.

Improving communication

The application does have a rather costly all-to-all communication, when dealing with the frequency domain Z direction of the dataset. One target was to overlap the communication, which can only be made by one CPU of the eight

CPU nodes, with meaningfull work, increasing the throughput. Another step is to use the SX-8 global memory option in MPI communication.

The application has the data domain decomposed in the Z direction. In the Z decomposition the Poisson equation is calculated, in an iterative manner using penta-diagonal solvers. Loops in the solvers and loops in general in the program are well vectorized. As the FFT transform needs to be solved a redistribution of the data has to take place. The FFT needs to have access to all values in the Z direction. The way it is made is using a new simple domain decomposition over the X dimension. In the X decomposition the spanwise direction is calculated using sine and cosine transforms. The transforms are implemented by FFTs using the NEC MathKeisan FFT library.

To change between these two representations, all data has to be redistributed between all MPI processes. In the earlier version the communication dealt with the full dataset at once. Instead of doing MPI communication and FFT of the whole data, first MPI communication is done on a Y layer. The communication works in blocks of data "decomposed" in the Y direction. By breaking down also the Y direction in independant blocks, see figure 7, it is possible to create a pipelined loop that deals with the different stages. This allows that one SMP thread can deal with the MPI communication, while the other SMP threads can do calculations in the other Y blocks. Thus effectively overlapping the communication done by one CPU with FFT calculation that is done with remaining CPUs. As the first layer is calculated, the next layer is transported. The first layer that already has been communicated can then be calculated on by the FFT algorithm and so on.

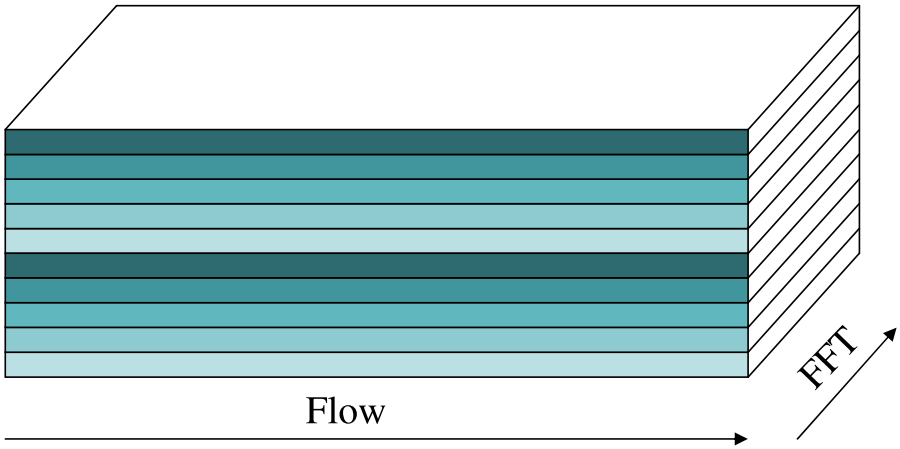

Fig. 7. Treating the Y dimensions as independant blocks in the communication and calculation of the sine-cosine transforms

The stages that need to be considered are

1. Reorganizing the data per CPU
2. Sending the data with `MPI_ALLTOALL`
3. Reorganizing the data for the FFT
   Calculating the FFT
   Reorganizing the data per CPU
4. Sending the data with `MPI_ALLTOALL`
5. Reorganizing the data into the original format

and each block will go through these stages.

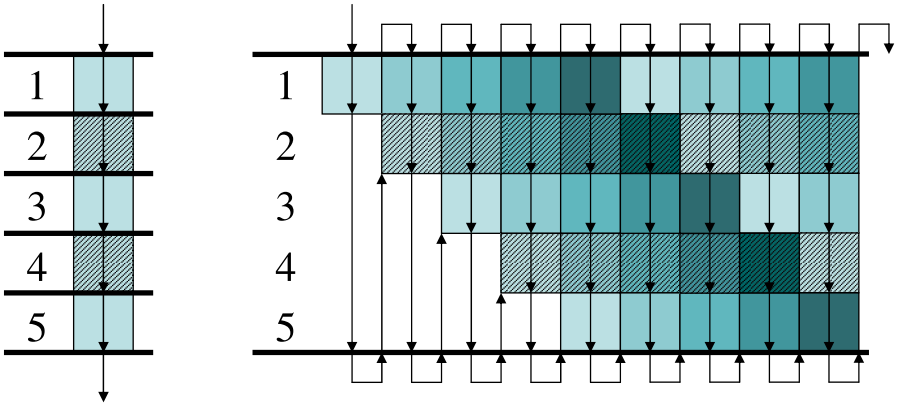

Fig. 8. Working through the blocks in a pipelined manner instead of like before the whole data at once gives less syncronization points between the threads.

The main target behind this new pipelined approach is to overlap the communication with work. As can be seen in figure 9, as the single threaded MPI call can be done in parallell, the rest of the CPUs do not have to sit idle. This advantage is only possible in a hybrid program MPI/SMP. As each MPI process can use up to 8 CPUs per node it means that up to 7 CPUs sit idle in the old version in some of the stages (2 and 4).

This work distribution improves scaling as the communication can overlap with the calculation. This is especially important as all data has to be redistributed between all MPI processes at every iteration.

Global Memory

The transport bufferes used in the `MPI_ALLTOALL` were placed in the global memory region with compiler directives. Global memory is normaly used by the MPI implementation, and by placing the data there from the program directly databuffer copy can be saved. Global memory regions are still memory local to the machine, therefore not inhibiting performance on the local level.

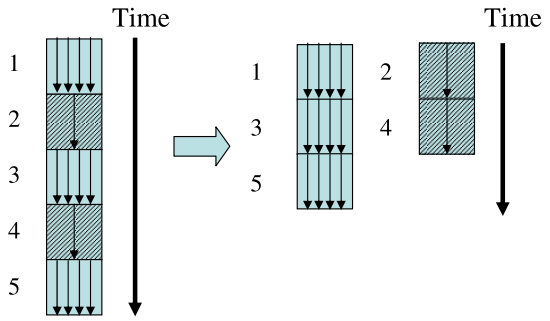

Fig. 9. Overlapping computation with communicaton makes the most out of the computer using the single threaded MPI implementation.

Asynchronous Communication

As the routines had been adapted for this division in Y, the step of changing the MPI communication was taken. A communication pattern using `MPI_Put` was introduced. This gives each thread the possibility to exchange data to global buffers. This more asynchronous communication can also be achieved on other platforms implementing the `MPI_ALLOC_MEM` call to allocate global memory. The `MPI_ALLOC_MEM` call on the SX allocates global memory, a memory region that is specially treated by the MPI implementation.

5 Results

The tuning steps brought different improvements, single cpu performance and scaling speedups.

The initial change, moving from homemade sine and cosinus transforms gave a direct improvement of 34% for the Z symmetric path and 22% for the Z non-symmetric path. As can be seen in figure 6 the performance change is not as visible as the change in realtime as the number of operations also goes down using the library FFT. The larger improvement in the symmetric path can be attributed to the symmetry in the frequency data. Less data is used to reach the result, giving the symmetric path an advantage over the non-symmetric path. Both versions are used in research depending on the problems investigated.

In the scaling step most concentration was put into the non-symmetric path that is the more complex setup. Also the tridiagonal solver was improved with some directives.

The current version shows good performance and scalability for large problems on the SX-8 system. While earlier research was made with smaller models and less nodes (1-2), todays research is demanding larger models. The current models in use have between 90 M and 996 M grid points. This will continue

to grow in the future. The primary target is to scale the X dimension to enable studies of different flow phenomena over larger distances.

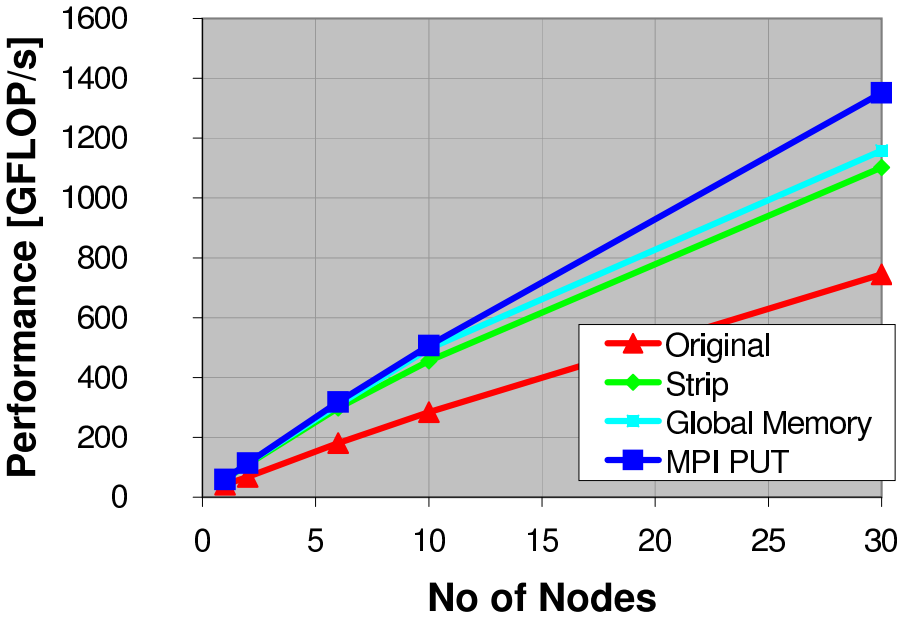

Fig. 10. Sustained performance for a case with 314 M grid points, detailing the different tuning stages.

Figure 10 shows the performance differencs between the different stages. The original performance on 30 nodes for the 314 M grid points dataset was 745 GFLOP/s. With all tunings it reaches 1.4 TFLOP/s on 30 nodes,i.e. it became almost twice as fast. The largest performance step is taken by the overlapping of communication with calculations (47 %), but also the use of the asynchronous communication improves scaling (16 %). The usage of global memory does only limit some memory copy but that is always helpful (5 %). The single node performance (only SMP without MPI), reaches 59 GFLOP/s and an efficiency of 46 %. Still the small case on 30 nodes retains an efficiency of 35 %. Less than 10 % drop off when compared to the first MPI measurement using 2 nodes at 44 %. Efficiency is calculated from a 16 GFLOP/s peak performance of one CPU.

As can be seen in the figure 11, the sustained performance on 70 nodes of NEC SX-8 at HLRS reaches 2.68 TFLOP/s using a large test case with 1100 M grid points. The strong scaling plot between 15 and 70 nodes, show an efficiency between 39 % and 30 %. 1.27 TFLOP/s was the performance of the original program using 70 nodes.

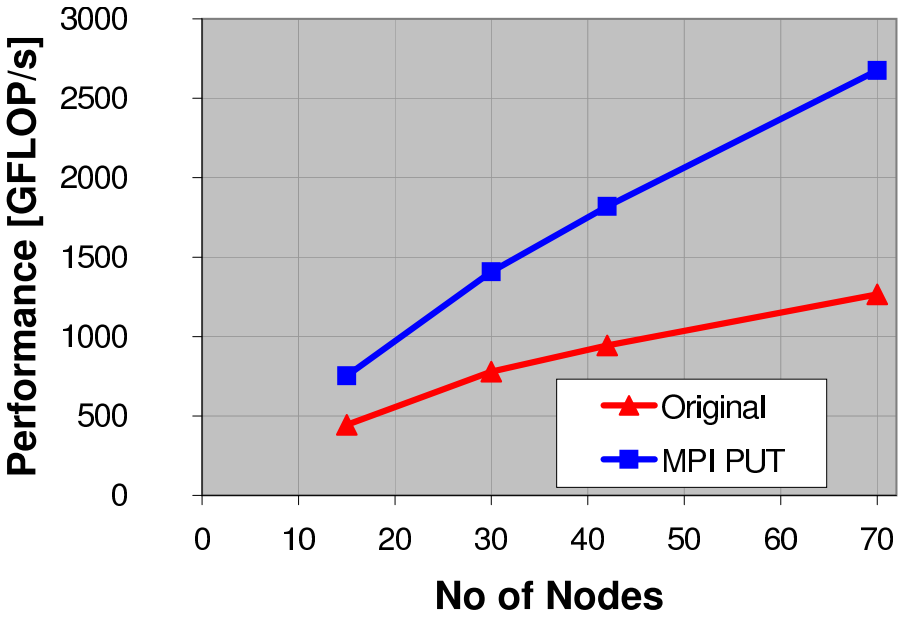

Fig. 11. Sustained performance for a case with 1100 M grid points, detailing the different tuning stages.

5.1 Computational Results from IAG

The numerical code has been executed on the NEC-SX 8 of the hww GmbH, Stuttgart. Using the original version the code attains 4.1 GFLOP/s of 16 GFLOP/s theoretical peak performance on a single processor at a vector operation ratio of 99% and an average vector length of 222. In runs on 14 nodes the code reaches 407 GFLOP/s with a RAM requirement of 193 GByte. The computation time is $0.9\mu s$ per time step and grid point on a computational grid of $2226\times793\times224$ (streamwise $\times$ wall-normal $\times$ spanwise) grid points.

References

[Bon99] Bonfigli, G., Kloker, M.: Spatial Navier-Stokes simulation of crossflow-induced transition in a 3-d boundary layer. In: Nitsche, W., Heinemann, H.-J.; Hilbig, R. (eds) New Results in Numerical and Experimental Fluid Dynamics II. Proc. 11. AG STAB/DGLR Symposium, NNFM 72. Vieweg Verlag, Braunschweig (1999)

[Mes04] Messing, R.: Direkte numerische Simulationen zur diskreten Absaugung in einer dreidimensionalen Grenzschichtströmung. PhD Thesis, Institut für Aero- und Gasdynamik, University of Stuttgart (2004)

[Sch06] Scholz, P.: Private Communication, Institut für Strömungsmechanik, TU Braunschweig (2006).

[Was02] Wassermann, P., Kloker, M.: Mechanisms and passive control of crossflow-vortex induced transition in a three-dimemsional boundary layer. J. Fluid Mech., **456**, 49–84 (2002)