V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal, 14–17 June 2010

HYBRID PARALLELIZATION OF A TURBOMACHINERY CFD CODE: PEFORMANCE ENHANCEMENTS ON MULTICORE ARCHITECTURES

Christian Simmendinger[1], Edmund Kügeler[2]

[1]T-Systems-Solution for Research
Pfaffenwaldring 38-40, 70569 Stuttgart, Germany
e-mail: Christian.Simmendinger@t-systems-sfr.com

[2]Institute of Propulsion Technology, German Aerospace Center (DLR)
Linder Höhe, D-51147 Cologne, Germany
e-mail: Edmund.Kuegeler@dlr.de

Key words: Fluid Dynamics, Turbomachinery, parallelization, multicore, hyperplane

Abstract. The CFD code TRACE used to be parallelized by means of a domain decomposition and MPI for distributed cluster architectures. The roadmap for x86 processor development however shows that the single core performance will grow only very moderately in the future. Instead one processor will integrate more and more cores. For the traditionally parallelized code this means to further increase the number of domains, which in turn has several disadvantages: communication overhead will increase as the ratio of ghost cells to inner cells in a CFD Code becomes more and more unfavorable, convergence rates will drop with decreasing local block sizes and load balancing will become a major challenge. In order to address these problems, a new hybrid parallelization has been implemented. The parallelization between domains is done with MPI, whereas the loop parallelization is done with phtreads. It now is possible to compute one domain on many cores without using additional domains.

1 INTRODUCTION

In the modern process of the aerodynamic design of multistage compressors and turbines for jet engines and stationary gas turbines, 3D-CFD plays a key role. Before building the first test rig several designs need to be investigated by means of numerical simulations. Large-scale computing resources here allow more detailed numerical investigations with bigger numerical configurations: up to 30 million points in one simulation is becoming a standard configuration in the industrial design. The designers require a completed computation of this configuration over night in order do analyze and improve the design throughout the day.

Whereas the computing resources are growing in terms of the amount of cores for each processor, the performance enhancements for a single core in contrast remain small. Many CFD codes have been parallelized using traditionally domain decomposition and MPI, but with a further splitting of the domains the ratio of volume to surface cells become more and more unfavorable and the communication overhead is growing significantly. The majority of CFD codes hence needs to be modified in order efficiently use the new multicore hardware.

The multicore paradigm also implies that bandwidth to main memory is shared between the available cores. In a first step towards a new implementation we have optimized the scalar performance of the Trace code. In doing so, we have achieved a scalar speedup of around 200% - depending on platform and use case. In order to reduce cache misses to a minimum a complete redesign of the data structures has been implemented.

The Trace code uses an implicit time integration scheme in combination with a dual time stepping approach to solve the unsteady Navier Stokes Equations. The system of equations then is solved using a Gauss-Seidel relaxation algorithm. The scalar efficiency of the forward and backward loop is enhanced using the hyperplane strategy [1]. This hyperplane also allows loop parallelization across the multi-core chips. The multicore parallelization is done for both, the structured and unstructured part of the Trace CFD-Code.

The performance enhancements are shown on standard applications for turbomachinery design, a multistage compressors and a high pressure turbine.

2 NOMENCLATURE

C	[J/(kg K)]	Specific heat capacity
e	[kg m/s^2]	Total specific energy
H	[m]	Height
M	[kg/s]	Mass flow rate
P	[Pa]	Pressure
Q	[-]	Vector of conservative variables
r	[m]	Radius
R	[J/(kg K)]	Specific gas constant
RPM	[min^{-1}]	Rotation speed
T	[K]	Temperature
u,v, w	[m/s]	Velocity components
ε	[kg m/s^2]	Internal specific energy
η	[-]	Efficiency

κ	[-]	Isentropic coefficient
ρ	[kg/m^3]	Density
Π	[-]	Pressure ratio

2.1 Subsripts

abs	Absolute
ADP	Aerodynamic design point
inlet	Inflow plane
is	Isentropic
p	Pressure
red	Reduced
stat	Static
tot	Total
v	Volume
1	Inlet
2	Outlet

3 SIMULATION SYSTEM

In this study we examine the CFD code TRACE, a three-dimensional, steady and unsteady flow solver for the Favre- & Reynolds-averaged compressible Navier Stokes equations. TRACE is a hybrid solver using structured as well as unstructured grids and a domain composition for parallelization of the computational domain. TRACE is focused on the physics of turbomachinery, and is integrated in the design process of the MTU Aero Engines AG. The structured and unstructured solver modules interact with a conservative hybrid-grid interfacing algorithm which allows using a mismatched abutting interface between structured and unstructured grid blocks [7], [15].

The numerical features of the hybrid-grid CFD solver are its second-order-accurate Roe's upwind spatial discretization to the convective fluxes with the MUSCL or linear reconstruction approaches, and its first- or second-order accurate implicit predictor corrector formulation [8]. Furthermore, it has a wide variety of models adapted for turbomachinery flows, e.g.

- Implicit steady and unsteady nonlinear solvers
- Implicit nonreflecting boundary conditions [14]
- Nonlinear aero elasticity module [10]
- Linearized solver in the frequency domain
- Two equation turbulence model, based on Wilcox k-ω model, special extension for rotating, compressible flows and streamline curvature [5]
- Multimode transition model [13]
- Higher Order discretization schemes
- Multifrequency Phase-Lag model [11], [12]

For more details of the TRACE-solver, please refer to the reference [4] and [8].

4 SCALAR OPTIMIZATIONS

In a first optimization loop the entire single core performance of the TRACE code has been improved by a factor of around 200%. In order to reduce cache misses to a minimum, a complete redesign of the C-language data structures has been implemented.

The original C data structures were implemented according to intensive or extensive properties of physical values and/or the mesh topologies (e.g. pressure, velocities,

volumes and coordinates or temperature). While this approach improves readability of the code, it also can have a negative impact on performance – especially if these data structures are becoming too large. We can easily identify two main factors for this performance impact: 1. All modern cpu's access the main memory in the form of cachelines (typically 32 -128 bytes, with a linear address space) 2. C data structures are allocated such that the elements of the structure are subsequent in memory. The combination of these two factors implies that any access to the first element of a data structures will retrieve all subsequent elements of this data structure – if required or not.

In order to improve the efficiency of the memory access, we hence have reorganized the data structures into "hot" and "cold" parts, depending on their frequency of use within the respective subroutines of the TRACE code. We give an example below, where the *FPhys* C data structure

```
typedef struct
{
   Ffloat          T, u, v, w;
   ...
   Ffloat          tke, tls;
   ...
} FPhys;
```

has been transformed into.

```
typedef struct
{
   Ffloat          T, u, v, w;
   ...
} FPhysFlow;

typedef struct
}
   Ffloat          tke, tls;
   ...
} FphysTurb;
```

We note that changes in these data structures affect almost the entire code base. The transformation hence required almost 50.000 changes to the implementation. In order to accomplish this complex and time consuming task, we have written a dedicated Perl src-src pre-processor. We emphasize that the changes did not affect readability of the code in a negative way.

After all changes were implemented, we observed a scalar speedup of 180% to 240% – depending on platform and use case.

5 THE HYPERPLANE METHOD

In order to further improve the scalar performance of TRACE we have implemented a modified hyperplane method. The hyperplane method itself goes back to a publication

of Leslie Lampart in 1972 [1] and was originally designed to permit parallelization for the ILLIAC IV for specific recursive loop structures.

```
DO 99 J = 2,M
DO 99 K= 2,N
      U(J,K)=(U(J+1,K)+U(J,K+1)+U(J-1,K)+U(J,K-1))*0.25
99  CONTINUE
```

Lampart observed that it is possible to parallelize this loop by means of a loop transformation which computes the array U in the form of hyperplanes. In these hyperplanes the elements of each hyperplane *I+J=const*. are independent of each other.

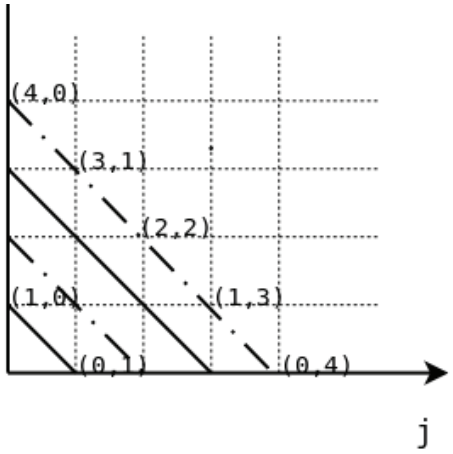


Figure 1: The hyperplane approach in cartesian coordinates.

We use a modification of this method: We have additionally restructured the access to the hyperplane such that this access pattern becomes stride 1. Instead of computing the loop in cartesian coordinates we here first use a transformation onto hyperplane coordinates. In the C language the above loop assumes the form:

```
for (hp =  0; hp < all_planes; hp++ )
      for (cnt = start[hp]; cnt < stop[hp]; cnt++ )
      {
            int mx0 = midx[cnt][0];
            int mx1 = midx[cnt][1];
            int px0  = pidx[cnt][0];
            int px1  = pidx[cnt][1];
            U[cnt] = (U[px0]+U[px1]+U[mx0]+U[mx1]) * 0.25;
       }
```

The arrays *pidx* and *midx* here point to the next and previous hyperplane, respectively. Our approach optimizes both spatial data and temporal data locality: All elements of the previous hyperplane can be used in the calculation of the next hyperplane. Cache thrashing is minimized.

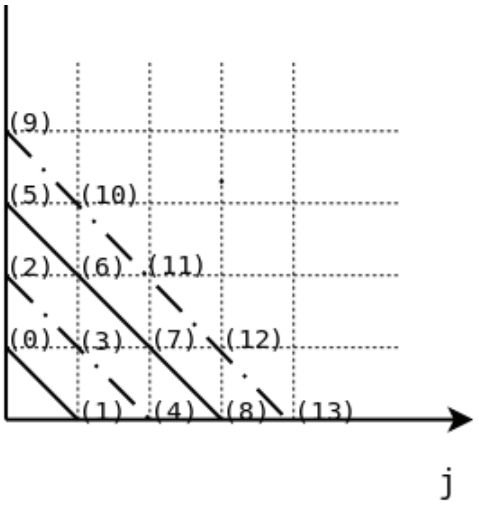


Figure 2: The hyperplane approach in hyperplane coordinates.

After having implemented all required changes we observed a speedup of more than two for the corresponding parts of the solver (forward and backward loop of the Gauss-Seidel relaxation algorithm).

6 HYBRID PARALLELISATION

For a block structured CFD solver such as TRACE, the largest block determines the maximal limit of scalability. While in principle it is possible to again further split this largest block into smaller parts it is often impractical, since smaller MPI domains come with a high overhead in terms of communication and decreased convergence rates for implicit solvers.

For these reasons and in order to make better use of current multi-core architectures, we have implemented a hybrid parallelization model, which is based on pthreads and MPI. In this implementation we use pthreads for all processing units of a CPU socket and bind the enveloping MPI process to the socket. We did a profiling run and subsequently have parallelized all loops which required one percent or more of CPU time. The multicore parallelization has been done for both, the structured and unstructured part of the Trace CFD-Code.

The speed up factor for a single block computing on four cores with the new parallelization (instead of using only one core with the previous implementation) is more than three.

7 HIGH PERFORMANCE CLUSTER

The Institute of Propulsion Technology uses two PC-Clusters provided by T-Systems-SfR. The first one is a 45 node cluster with Intel Harpertown series 5440 quadcore chips. Each node consists of two processors and 16 GB RAM. Interconnect is an Infiniband double rate data network.

The second one is a 200 node cluster with Intel Nehalem series 5540 quadcore chips, figure 3. Here each node consists of two processors and 24 GB RAM. Interconnect is also an Infiniband double rate data network. The second cluster will be used for the benchmarks of the hybrid parallelization.

Figure 3: Nehalem cluster of the Institute of Propulsion Technology

8 TESTCASES AND RESULTS

8.1 Generic Testcase – Channel Flow

The first testcase is a generic channel flow. The mesh is shown in figure 4. It consists of eight structured blocks of equal size. The purpose was to have a constant load for each process when testing the parallelization. The measurement of the performance of the hybrid parallelization can be very easily compared between different numbers of MPI-processes combined with different numbers of used threads on a multicore chip.

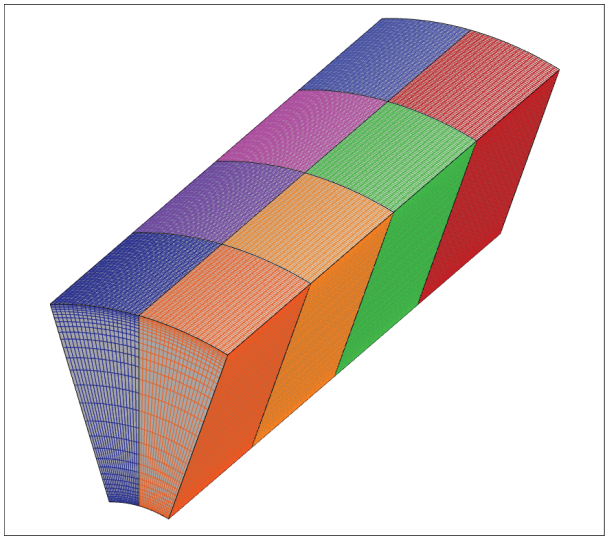


Figure 4: Block structured mesh of the channel testcase

Each block has approximately 535.000 cells, so the whole configuration has about 4 Million cells. The simulation is done in a steady mode, using the implicit algorithm, the

k-ω turbulence model with wall functions and nonreflecting boundary conditions at inlet and exit plane.

The simulations were done on the above mentioned cluster with the following variations:

Using 1 node, 8 MPI-Processes, 1 thread per process
Using 1 node, 2 MPI-processes, 4 threads per process
Using 1 node, 2 MPI-processes, 8 threads per process (Hyperthreading)
Using 2 nodes, 4 MPI-processes, 4 threads per process
Using 2 nodes, 4 MPI-processes, 8 threads per process (Hyperthreading)
Using 4 nodes, 8 MPI-processes, 4 threads per process
Using 4 nodes, 8 MPI-processes, 8 threads per process (Hyperthreading)
The results are shown in table 1.

Testcase	Compiler- /MPI-version	No of nodes	No of MPI-processes	No of threads per process	Runtime
Channel	icc11.1 /openmpi	1	8	1	01:25:29
Channel	icc11.1 /openmpi	1	2	4	01:31:44
Channel	icc11.1 /openmpi	1	2	8	01:58:52
Channel	icc11.1 /openmpi	2	4	4	01:28:16
Channel	icc11.1 /openmpi	2	4	8	01:01:42
Channel	icc11.1 /openmpi	4	8	4	00:35:23
Channel	icc11.1 /openmpi	4	8	8	00:31:13

Table 1: Overview of hybrid calculations for the generic testcase

The first run is the original mode to run TRACE. The process is parallelized only via MPI, so each core has to compute one block. This was the fastest way for the original code without further splitting the mesh. The second run uses the hybrid parallelization, each process has to compute 4 blocks, but each is computed parallel on 4 cores. The second run needs less than 10% more runtime. The reason therefore is that not all loops are parallelized using the phtreads. All boundary condition loops for example are not parallelized. The interesting runs are number four to seven. Now we are able to use more cores than blocks the mesh consist of. Due to this fact we have a speed up of 1.4 for 4 MPI-Processes and a speedup of nearly 3.0 for 8 MPI-processes, which is the maximum speedup for this testcase, each block is computed in one process using 4 cores.

8.2 Multistage Compressor

The testcase is a 15 stage compressor with an additional inlet guide vane and an outlet guide vane, figure 5. It was designed by MTU Aero Engines for a stationary gas turbine while for rig testing a scaled variant was used.

The compressor has five transonic front stages, while the remaining stages are subsonic. The tip clearances of the rotors are about 1 mm; all stators are modeled cantilevered with a hub clearance also about 1 mm. One bleed is installed after stage 4, where approximately 3.5% of the inlet mass flow is blown off.

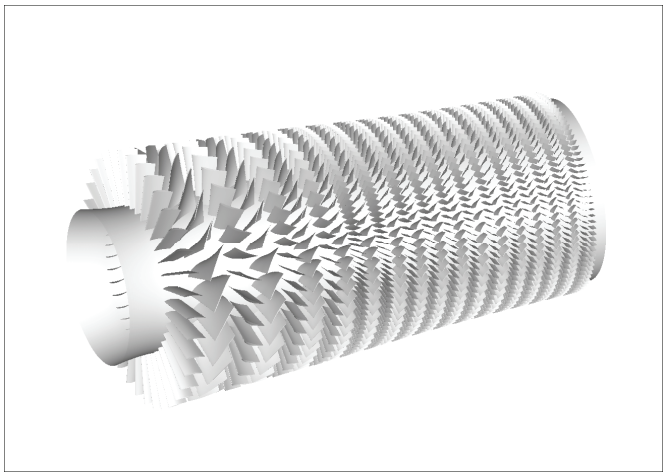


Figure 5: Configuration of the compressor

The inlet absolute total pressure is app. P_{tot} = 60.000 Pa, the inlet absolute total temperature app. T_{tot} = 298 K, the simulations were done for the 100% speedline (RPM = 9230 min^{-1}).

The mesh for the whole compressor is generated using the grid generator G3DMESH, which is developed by CFD-Norway and extended for tubomachinery by the Institute of Propulsion Technology. It uses the technology of parameterized templates, which defines the topology of the structured mesh, figure 6. More detailed information can be found in Weber [6].

The mesh consists of 19,11 million grid points overall for 32 blade rows, the radial resolution is 65 points and the tip clearances have a spanwise resolution of 7 points. Normal to the blade surfaces we have a Low Re resolution with y^+ ranging from 1 to 2.5, at the endwalls the y^+ values varies from 25 to 50.

For the simulation we use TRACE with a steady implicit algorithm, a second order Roe scheme with MUSCL extrapolation and the two equation k-ω turbulence model with extensions for rotational and compressible effects, see Kozulovic et al. [5].

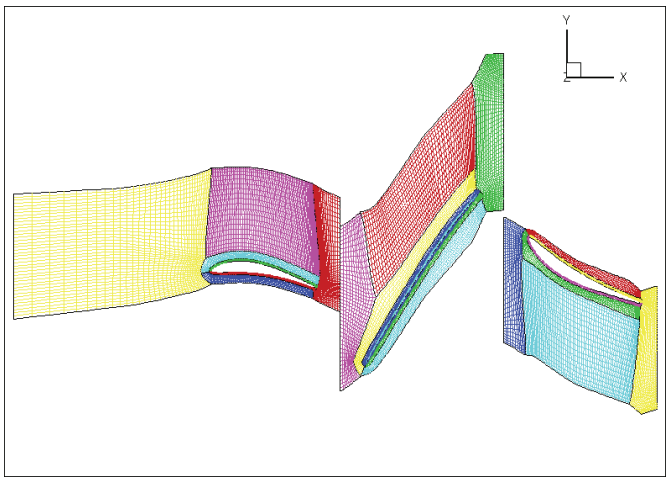


Figure 6: Structured mesh at tip for IGV, Rotor 1 and Stator 1 (OCH topology)

For the inflow, outflow and mixing planes we use nonreflecting boundary conditions with a Fourier decomposition formulated in the relative frame of reference.

The compressor was numerically investigated extensively concerning the influence of real gas formulations in the code [16] and modelling the nearly the real geometry with fillets at blades and vanes [2]. Figure 7 shows the compressor map for the case without fillets and the comparison of simulations with an ideal gas and a real gas formulation in the TRACE code.

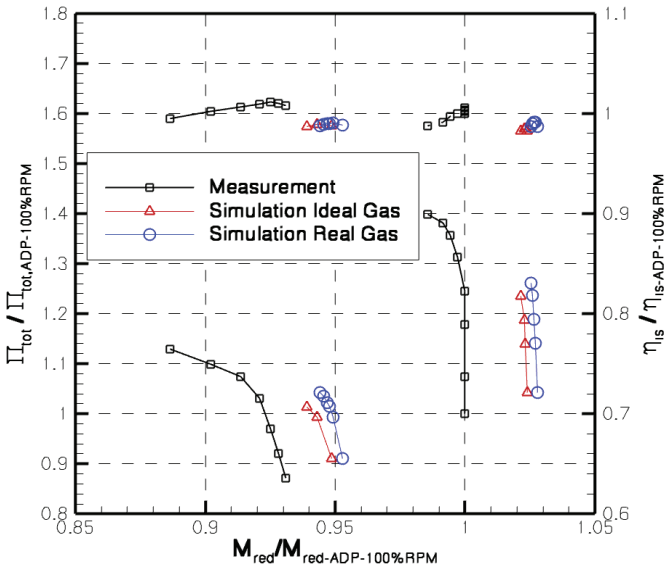


Figure 7: Compressor map

The compressor map shows two speedlines, the nomimal speedline at 100% RPM and the 95% speedline. One speedline needs up to five operationg points to analyse the interaction of all stages in the aerodynamics of the compressor. All calculations along the speedline (operating points) were restarted on the previous solution exept the first one beginning with (lowest back pressure). One calculation needs about 5000 Iterations to converge.

Testcase	Compiler- /MPI-version	No of nodes	No of MPI-processes	No of threads per process	Runtime
Compr.	icc11.1 /openmpi	8	60	1	06:45:30
Compr.	icc11.1 /openmpi	30	60	4	02:56:16
Compr.	icc11.1 /openmpi	30	60	8	02:31:46

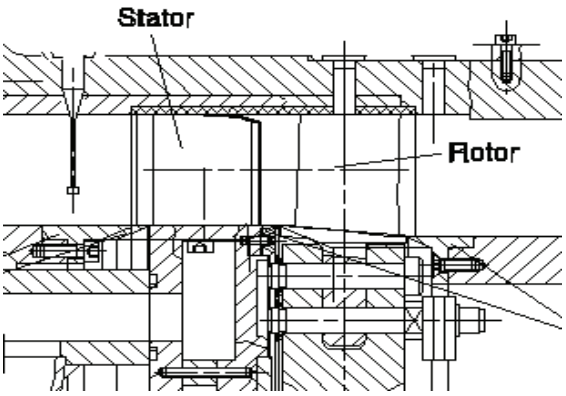
Table 2: Runtime comparison for the compressor testcase

Table 2 shows the comparison of runtime for one operating point of the map. The simulation with parallelization using only MPI needs about 2,2 more time than the simulation using the hybrid parallelization and 4 threads, respectively 2,5 than the simulation using 8 threads and Hyperthreading.

The goal is to simulate the whole compressor map over night during the design phase. Computing the speedline in parallel now it becomes possible.

8.3 Testcase Single Stage Turbine

The third testcase is a transonic, high pressure, single stage turbine taken from the European research project ADTurB. The stage was measured at the turbine test facility at DLR in Goettingen. For the study in this paper the large gap/rigid rotor configuration without vane trailing edge blowing has been chosen [21]. A cross view of the geometry as well as the basic geometrical parameters are summarized in figure 8.



blades	43	64
stagger angle	51.9°	32.7°
hub radius	0.238 m	0.235 m
tip radius	0.274 m	0.274 m
chord length	0.049 m	0.033 m
aspect ratio	0.721	1.166
pitch/chord ratio	0.749	0.747

Figure 8 : Experimental setup at the experimental turbine facility RGG in Goettingen and geometrical data of the ADTurB turbine stage Rehder [21].

A transonic operating point of the stage at the rotational speed of RPM = 6957 min^{-1} (77.8% of the nominal speed) was calculated. The inlet conditions were T_{t0} = 311.0 K and P_{t0} = 131865.0 Pa. The inflow to the vane is axial (no swirl), all values were measured in the experiments. In the inlet plane measured total pressure profile were imposed at hub and tip.

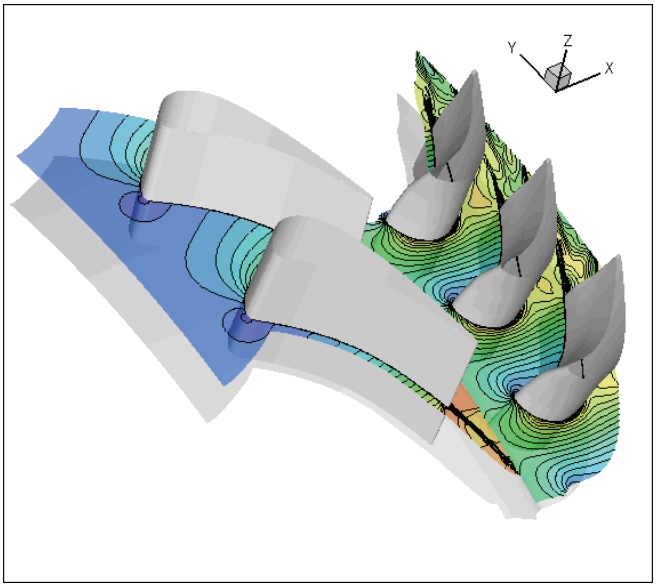


Figure 9: Mach number at midspan of the ADTurB turbine

Figure 9 shows the configuration simulated with two stator vanes and the three rotor blades and the Mach number distribution at midspan for the steady state calculation.

The global stage performance parameters in terms of mass flow, pressure ratio, and isentropic efficiency are summarized in table 3.

	experiment	TRACE
exit pressure p_2	**40219.0 Pa**	
massflow [kg/s]	**4.73**	**4.63**
pressure ratio pt0/pt2	**2.72**	**2.81**
efficiency [%]		**88.10**

Table 3: Global performance data (exp. data taken from Rehder [21])

The stage consists originally of 43 stator vanes and 63 rotor blades. For unsteady calculations the system can be scaled to 2 stator vanes to 3 rotor blades to achieve direct periodicity, which will be equivalent to 42 stator vanes and 63 rotor blades for the full ring. The mesh consists of 127 structured blocks and 28,24 Million points overall. The TRACE solver was configured for the case in the following manner:

- Steady, 2nd Order MUSCL Upwind, van Albada Limiter
- Implicit Predictor-Corrector Algorithm
- Fully turbulent kω-model with KatoLaunder extension
- Entry, Exit, Interface: non reflecting BC's according to Giles
- All wall LowRe viscous boundary conditions

During benchmarking of the hybrid parallelization a steady state calculation from initialization until convergency was done. All 127 blocks have to be distrubuted to the processors/cores. In a real configuration the mesh size varies from 152.785 cells for the smallest block to 292.215 cells for the larger one. In order to achieve a good load for all cores, each processor/core should have to compute the same number of cells. Table 4 shows the average number of cells for each process and a Loadbalancing number for different number of processes, which describes the ratio of the number of cells between the process with the lowest load to the one with the highest.

Testcase	No. of processes	Max. number of points/process	„Loadbalancing" [%]
ADTurB	32	844800	80,90
ADTurB	40	768000	81,33
ADTurB	64	424960	66,87
ADTurB	127	292215	50,45

Table 4: "Loadbalancing" for the ADTurB Testcase

A good and acceptable "Loadbalancing" for the testcase are 40 processes. With the origin Trace this means we can use only 5 nodes at the cluster. With the new hybrid parallelization now we can use 20 nodes.The speedup for thr new code is shown in table 5. The runtime was reduced to 39%, the speedup factor is 2.6.

Testcase	Compiler- /MPI-version	No of nodes	No of MPI-processes	No of threads per process	Runtime
ADTurb	icc11.1 /openmpi	5	40	1	09:15.42
ADTurb	icc11.1 /openmpi	20	40	4	03:56:15
ADTurb	icc11.1 /openmpi	20	40	8	03:35:01

Table 5: Runtime comparison for the ADTurB testcase

The convergence of the solution alogrithm should not be influenced. Figure 10 shows the comparison of the convergency history for the three caclulations which are identically.

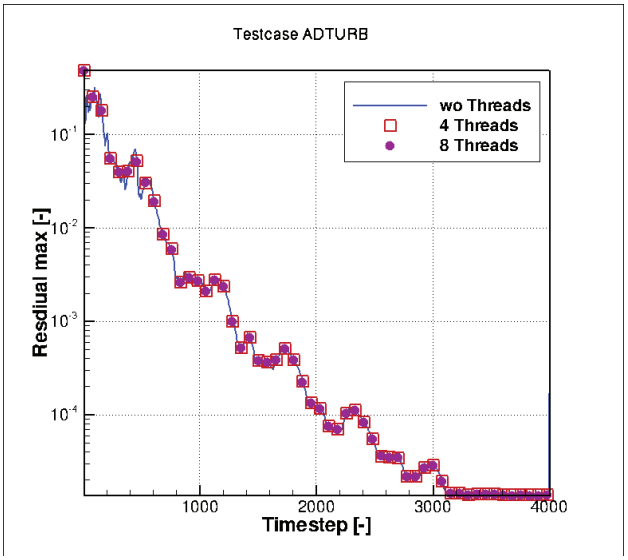


Figure 10: Convergency of the ADTurB Testcase

9 CONCLUSIONS

A hybrid parallelization using MPI for the distributed computing across nodes of a cluster and pthreads for the parallelization across the cores on a processor is developed for a CFD code. The code is parallelized by domain decomposition using MPI, whereas pthreads are used for loop parallelization. The successful implementation shows a significant speedup when using more cores than domains existing in the mesh. It avoids larger communication overhead through extensive splitting of the computational domain.

The speedup for the generic testcase shows nearly 75% of the theoretical value, whereas in real application it drops down to 65%. The reason therefore is not yet understood and part of further investigations.

10 ACKNOWLEDGMENTS

The authors thank MTU Aero Engines for providing the test cases and the measurement data for the whole compressor.

The work is done within the project HI-CFD (High Efficient Implementation of CFD-Codes for HPC-Many-Core-Architectures), which is funded by the Federal Ministry of Education and Research (BMBF) of Germany.

REFERENCES

[1] Leslie Lamport: The Parallel Execution of DO Loops. **Commun. ACM** 17(2): 83-93 (1974)

[2] E. Kügeler, D. Nürnberger, A. Weber, and K. Engel, Influence of blade fillets on the performance of a 15 stage gas turbine compressor. *ASME Turbo Expo.* **GT2008-50748** (2008)

[3] Burcat, A.,Branko, R., 2005, "Third Millennium Ideal Gas and Condensed Phase Thermochemical Database for Combustion with Updates from Active Thermochemical Tables", University of Chicago, Israel Institute of Technology.

[4] Franke, M, Kügeler, E., Nürnberger, D.,2005, „Das DLR-Verfahren TRACE: Moderne Simulationstechniken für Turbomaschinenströmungen", **DGLR-2005-211, Friedrichshafen**, 26.-29. September.

[5] Kozulovic, D.; Röber, T. K.; Kügeler, E.; Nürnberger, D.,2004, "Modifications of a Two-Equation Turbulence Model for Turbomachinery Fluid Flows", **DGLR [Hrsg.]: DGLR Jahrbuch, DGLR, Deutscher Luft- und Raumfahrtkongress,** Dresden.

[6] Weber, A., 2006, „3D Structured Grids for Multistage Turbomachinery Applications based on G3DMESH", **DLR-Bericht IB-325-03-06, Deutsches Zentrum für Luft- und Raumfahrt,** Köln.

[7] Yang, H., Nürnberger, D., Kersken, H.-P.: "Toward Excellence in Turbomachinery Computational Fluid Dynamics, 2006, "A Hybrid Structured-Unstructured Reynolds-Averaged Navier-Stokes Solver", **ASME Transactions, Journal of Turbomachinery, pp. 390-402, Vol. 128,** 2006.

[8] Kügeler, E., 2005 Numerisches Verfahren zur genauen Analyse der Kühleffektivität filmgekühlter Turbinenschaufeln, **DLR Forschungsbericht 2005-11,** Köln.

[9] Nürnberger, D., Eulitz, F. Schmitt, S., Zachcial, A., 2001, "Recent progress in the Numerical Simulation of unsteady viscous multistage turbomachinery flow", **ISOABE 2001-1081,** Bangalore, September.

[10] Schmitt, S., Eulitz F., Nürnberger, D., Carstens, V., Belz, J.,2002, "Simulation of Propfan Forced Response using a Direct Fluid-Structure Coupling Method", **4th European Conference of Turbomachinery, Fluid Dynamics and Thermodynamics**, Florenz.

[11] Schnell, R., 2001, "Experimental and Numerical Investigation of Blade Pressure Fluctuations on a CFK-Bladed Counterrotating Propfan". **ASME Turbo Expo Land, Sea and Air, New Orleans, Louisiana,** USA, June 4-7.

[12] Schnell, R., 2004, "Investigation of the Tonal Acoustic Field of a Transonic Fan Stage by Time-Domain CFD Calculations with Arbitrary Blade Counts", **ASME-Paper 2004-GT-54216**.

[13] Kozulovic, D., Röber, T., Nürnberger, D., 2007, "Application of a multimode transition model to turbomachinery flows", **Proc. 7th European Turbomachinery Conference, pp 1369-1378**, Athens, Greece.

[14] Ashcroft, G., Schulz, J.,c2004, "Numerical Modeling of Wake-Jet Interaction with Application to Active Noise Control in Turbomachinery", **AIAA Paper 2004-2853**.

[15] Yang, H., Nürnberger, D., Nicke, E., Weber, A., 2003, "Numerical Investigation of Casing Treatment Mechanisms with a Conservative Mixed-Cell Approach" **ASME-Paper 2003-GT-38483**.

[16] Kügeler, E., Fakhari, K., Mönig, R., "Influence of Real Gas Phenomena on a 15 Stage Gas Turbine Compressor", The 12th International Symposium on Transport Phenomena and Dynamics of Rotating Machinery Honolulu, Hawaii, February 17-22, 2008, **ISROMAC12-2008-20029**

[17] Zachcial, A., Nürnberger, D., "A Numerical Study on the Influence of Vane-Blade Spacing on a Compressor Stage at Sub- and Transonic Operating Conditions", **ASME-Paper 2003-GT-38020**, 2003

[18] Zachcial, A., Nürnberger, D., Kügeler, E., "Kopplungstechniken zur Simulation vielstufiger Axialverdichter", **DGLR-2004-224**, Dresden, 20.-23. September, 2004

[19] Belrami, T., Galpin, P., Braune, A., Corenlius, C., "CFD Analysis of 15 Stage Axial Compressore Part I: Methods", **ASME-Paper GT 2005-68261**

[20] Belrami, T., Galpin, P., Braune, A., Corenlius, C., "CFD Analysis of 15 Stage Axial Compressore Part II: Results", **ASME-Paper GT 2005-68262**

[21] Rehder, H.-J.: *High Pressure Turbine Stage Configurations for the Brite/EuRam Project ADTurB (Geometry, Operating Points)*, **DLR IB 223-2000 A 08**, May 2003