A Java Based High Performance Solver for Hierarchical Parallel Computer Architectures

Thorsten Ludewig[1], Jochem Häuser[2], Torsten Gollnick[3] and Wuye Dai[4]
University of Applied Sciences Wolfenbuettel, Salzgitter, Germany

and

Hans-Georg Paap[5]
HPC Consultant, Barbing, Germany

Keywords: **Hierarchical parallel computer architecture, Java HPC, client-server computation, OOP, Internet-based computing, Internet-based data access, diverse scientific and engineering disciplines, collaborative engineering, portable HPC and geometry framework, legacy code integration, architecture independence, HPC without libraries, complex 3D geometries, just in time solver, Java Performance.**

I. Introduction

IN [1] the Java Ultra Simulator Technology (***JUST***) solver was presented, whose parallel strategy is based on the Java thread concept. While the thread concept work well with an SMP (Symmetric MultiProcessor) architecture, it cannot be applied to hybrid parallel systems, comprising nodes with distributed memory as well as multiple processors per node sharing memory. This type of architecture is hitting the market right now, for instance SUN`s new Fire V20Z, based on Athlon 64 bit technology, and, because of the excellent price/performance ratio, will gain wide acceptance in scientific computing. In addition, as was demonstrated using a Java test suite [1, 5, 6], substantial progress has been made over the last three years in Java's numerical performance as well as parallel efficiency. These test cases (e.g. matrix multiplication, Mandelbrot set or a Laplace solver) were utilized to perform almost one-to-one source codes comparisons,between Java and C++.
In this paper we will present results and performance comparisons for real CFD simulations of complex 2D and 3D geometries, using the two components of ***JUST***, ***JUSTGRID*** and ***JUSTSOLVER,*** on several different parallel computer architectures and will provide guidelines to achieving best efficiency from modern Java virtual machines (JVM). Second, a strategy will be devised to obtain automatic parallelization on modern hybrid parallel architectures. Furthermore, the layered software design will be presented.

JUSTGRID is a completely Java based software environment for the the user/developer of HPC software. ***JUSTGRID*** takes care of the difficult tasks of handling very complex geometries (aircraft, spacecraft, biological cells, semiconductor devices, turbines, cars, ships etc.) and the parallelization of the simulation code as well as its implementation on the internet. ***JUSTGRID*** builds the computational Grid, and provides both the geometry layer and parallel layer as well as an interface to attach any arbitrary solver package to it. ***JUSTSOLVER*** is a pure Java CFD solver plugin for ***JUSTGRID***, based on finite volume technique, and thus can be used for any kind of hyperbolic problem (system of hyperbolic equations).

[1] Central Systems and IT Division Head, Computing Center, Univ. of. Applied Sci., AIAA Member
[2] CEO, HPCC-Space, AIAA Member
[3] Senior Scientist, Univ. of. Applied Sci.
[4] Senior Scientist, Univ. of. Applied Sci.
[5] HPC Consultant, Barbing

II. Java Virtual Machine Performance Progress

All In [1][5][6] it was demonstrated that the single-processor performance of Java is on par with C++ and the speedup on common SMP machines is linear. In most cases, Java performs better than C++ binaries compiled with the well known and widely used GNU C Compiler (GCC). Only the high optimizing compilers (e.g. Sun ONE Studio Compiler for SPARC Solaris or the Intel Compiler for Linux) are able to create binaries with the same or sometimes slightly better performance. On an AMD Opteron (AMD64) Processor dealing with the latest Java Runtime Engine 1.5.0 the achieved single processor speed running a 40 time 40 Matrix multiplication is about 65% of a C code with *static* matrix dimensions. But in the common way of use with *variable* (dynamic) matrix dimensions the Java Runtime Engine is about *2 times faster* than a GNU C++ 64 Bit binary with all AMD64 optimization compiler flags enabled.

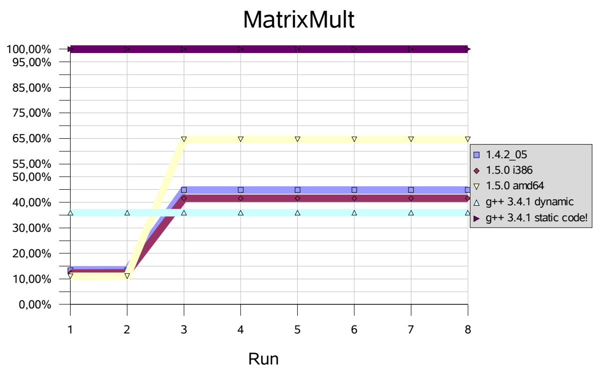

Illustration 1 *Results for a sequential matrix multiplication for a 40 times 40 matrix doing 10.000 iterations showing relations of variable java and c++ to static c++ code.*

III. Parallelization Issues for SMP and Hybrid Parallel Architectures

At present the parallelization of ***JUST*** is based on the Java *thread* concept. This *thread* concept has substantial advantages over the *PVM* or *MPI* library parallelization approach, since it is part of the Java language. Hence, no additional parallelization libraries are needed. Depending on the *thread* implementation of the computer's operating system (OS) and the size of the numerical problem, **dynamic load balancing** on SMP hardware architectures *is automatically achieved.* Starting with the end of 2004 AMD, Intel,Sun and other are delivering Dual Core Chips with two CPUs in one processor. In 2005/2006 Sun will deliver their CMT processors (CPU Multi Threading) with 4 hardware threads per core and 4 cores per processor. With this Processor an application can get the advantage of using 16 hardware threads concurrently. In the near future we will see many new SMP like systems with the power of lots of hardware threads and Java is the only environment that do not need any additional libraries to take advantage of these processors.

However, this concept is not sufficient for the new hierarchical/cluster architecture, and therefore needs to be extended.

A. Using Multi-Stage Parallelism

Today platforms for high-performance applications are more and more based on systems derived from the PC industries, that is off the shelf boards are used, equipped with several CPUs. Moreover, the CPUs are designed as multi-core platforms able to support real parallel streams of instructions. These powerful computers are then connected with a network to a so called cluster to provide a stronger parallel-working system.

To take advantage of these systems, the hierarchy of parallelism has to take into account. The fine-grained parallelism is located on the single computer where multi-core multi-processor systems are offered. Here the Operating System (OS) is responsible for providing a mechanism to exploit the hardware. On top of the OS the Java Virtual Machine (JVM) is adding convenience for the programmer for designing parallel working programs, that is, the paradigm for parallelism (Java Threads) provided by the JVM must be used.

As soon as the software should also using resources connected by a network, another strategy for a coarse-grained parallelism must be developed.

B. Fine-Grained Parallelism

The grid provided by a grid generator is converted into data format for the solver, either unstructured or bloc-structured (or both). In case of a block structured grid there exists an inherent parallelism on the block level. Additionally automatic load balancing using recursive bisection will be performed, based on domain decomposition, one block per thread. Depending on block size it is also possible to decompose a block into smaller data sets. Since every active component (multi-processor/multi-core) has access to the same memory, latency is not an issue in the handling of multiple threads.

C. Coarse-Grained Parallelism

Every resource outside of a node or even a computer can be another single computer or another cluster. Depending on the structure of a network information speed varies.

These delays must be taken into account in the development of a parallelization strategy. To avoid severe communication overhead each group of blocks being distributed to other computers needs to have a minimum number of direct neighbors to exchange data with. This is also achievable by applying recursive bisection..

For setting up the network-connected computer system for a coarse-grained parallel architecture, there are two possibilities.

The first strategy requires knowledge of the current state of the system including the properties of every single node. These properties are at least the number and the speed of processors and the amount of memory. Maybe the type of network connection is also of interest.

The second way requires a self knowledge of every node. This information must be offered to the whole system. By using **JINI**, a package provided by Sun Microsystems for network-centric services, every node acts as a service provider to all other nodes. The system is able to recognize the topology of itself and can establish high-performance connections between the nodes. The technology used for data exchange can be socked based or can use the New I/O (NIO) package, designed for high performance scalable network and file I/O.

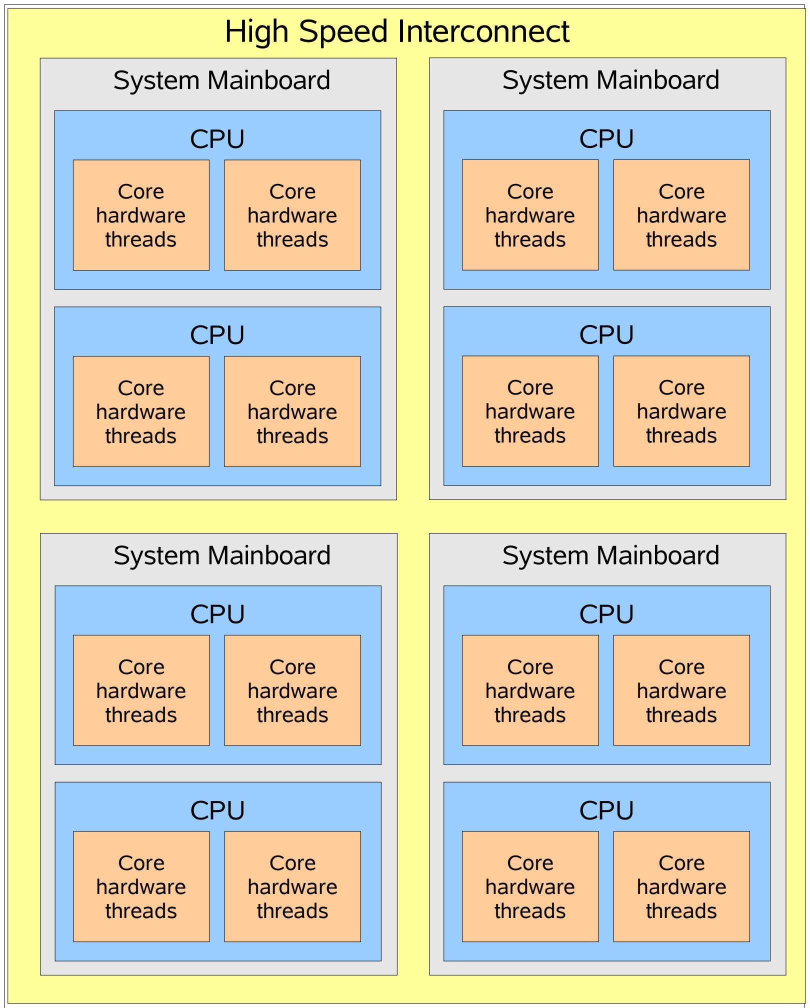

Illustration 2 *Scheme diagram of a hiracical topology fort HPCC*

IV. JUSTGRID

JUSTGRID is the basis for ***JUST*** (Java Ultra Simulator Technology) that is a revolutionary computing software that dramatically improves the ability to quickly create new kinds of software systems across the whole field of science and engineering, embedded in an Internet-based environment.

JUSTGRID is a completely Java based software environment for the the user/developer of HPC software. ***JUSTGRID*** takes care of the difficult tasks of handling very complex geometries (aircraft, spacecraft, biological cells,

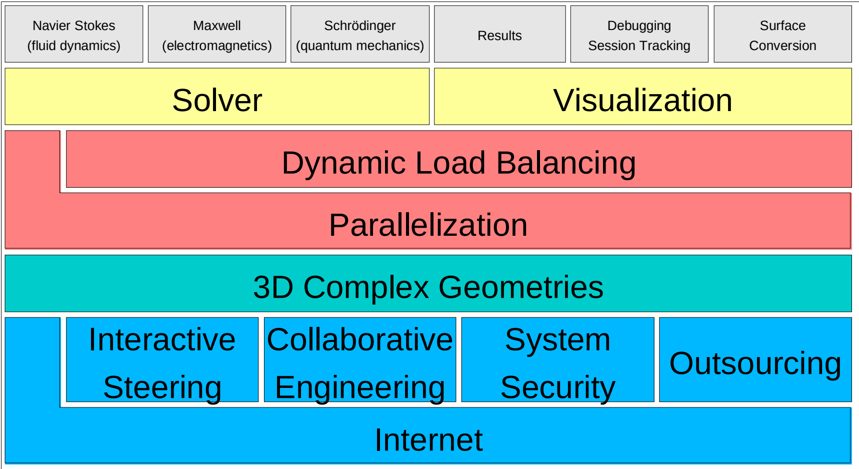

Illustration 3 *JUSTGrid a framework for HPCC in engineering, science and life sciences.*

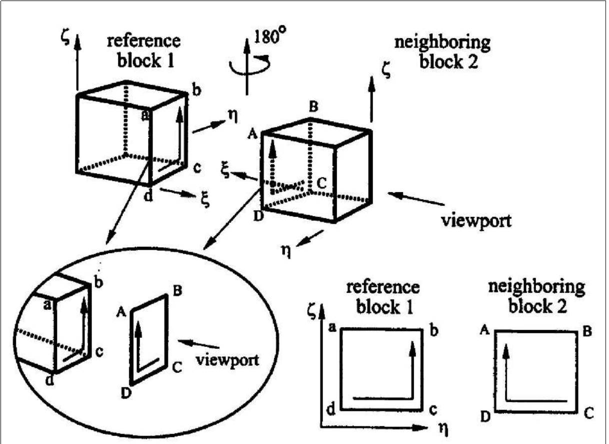

Illustration 4 *Determination of orientation of faces between neighboring blocks as seen from block 1 (reference block). The reference block is always oriented as shown and then the corresponding orientation of the neighboring face is determined (see Illustration 5)*

semiconductor devices, turbines, cars, ships etc.) and the parallelization of the simulation code as well as its implementation on the Internet. ***JUSTGRID*** builds the computational Grid, and provides both the *geometry layer* and *parallel layer* as well as an interface to attach any arbitrary solver package to it. See [1] for more information about the structure and architecture of ***JUSTGRID***.

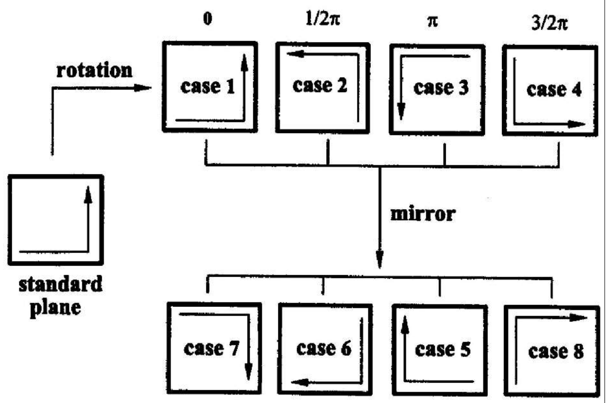

Illustration 5 *The 8 possible orientations of neigboring faces are shown. Case 1 to 4 are obtained by successive rotations. The same situation holds for cases 5 to 8 upon being mirrored.*

D. *JUSTGRID* initial tasks

1) Loading and parsing grid, topology, command and boundary condition files. ***JUSTGRID*** is able to load Plot3D, GridPro, Tecplot and XML grid files, topology, command and boundary condition files are simple ASCII text files in ParNSS format. (ParNSS is our legacy flow solver written in C).
2) Determine block face connectivity and orientation. ***JUSTGRID*** detects all matching block faces and their orientation (rotation) in one run for every block from the just loaded grid. [15]
3) Initialize block cells. ***JUSTGRID*** constructs a new instance of the selected cell implementation for each grid cell in a block.
4) Compute cell metrics. As the last initialization task ***JUSTGRID*** computes the normal vectors for each cell face and the finite volumes for each cell.

The time consumption in seconds for each initialization task is shown in tables 1 and 2 running on a Pentium 4 Mobile, 2GHz using Mandrakelinux 9.2. As a native compiled reference we select Amtec Tecplot 9.2 only loading the same grid. Comparing the two tables we see that with the 274 blocks grid the server Java Runtime Engine (Hotspot) is slightly slower as the client JRE but with the 1422 block grid the parsing time is less than 50% of the client JRE. This behavior can be often recognized in many different parts of Java applications. Only if the CPU workload is high enough the server JRE (Hotspot Engine) is able to find the CPU load intensive parts of the code and optimize these so called Hotspots on the fly during the code is executed.

Spacecraft, 274 blocks, 342362 vertices, 256268 cells, 18.2 MB file size

Runtime	Parsing	Connectivity	Initialize	Cell Metrics	Total
1.4.2_06 client	4,98	0,13	1,13	2,43	8,77
1.4.2_06 server	6,37	0,53	1,11	2,49	10,68
1.5.0 client	4,70	0,12	1,06	2,23	8,52
1.5.0 server	5,75	1,33	1,02	2,31	10,59
Tecplot 9					25,00

Table 1 *Initial task times in seconds for a 274 block grid*

Aircraft, 1422 blocks, 2251694 vertices, 170832 cells, 123.1 MB file size

Runtime	Parsing	Connectivity	Initialize	Cell Metrics	Total
1.4.2_06 client	61,04	3,24	6,24	16,53	87,62
1.4.2_06 server	28,08	3,22	6,00	15,88	53,64
1.5.0 server	24,38	2,93	4,91	14,18	46,61
1.6.0 EA b12 server	24,54	3,14	4,93	14,06	47,19
Tecplot 9					268,00

E. *JUSTGrid* run session

After the initialization tasks are finished ***JUSTGrid*** starts the computation session. The ***JUSTGrid*** session creates one solver plugin instance and one computation thread per block. While the computation is running a monitor thread automatically detects dead locks and gives always the current status of the running sessions. The monitor thread is also responsible for storing the final result (e.g. Tecplot format) after the computation becomes ready.

V. JUSTSolver

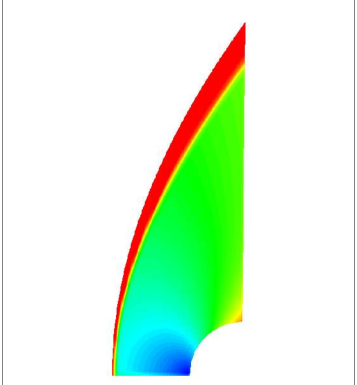
Illustration 6 *2D sample with JUSTSolver, 4000 iteration, AoA 0.0, Mach 2.0*

JUSTSolver is a sample implementation of a 2D and 3D Euler CFD solver, a plugin for ***JUSTGrid*** implementing the integral form of conservation laws. At this time we are validating the correctness of the solver results so we use only simple and well known samples.

F. Integral Form of Conservation Laws

In the following, the general case of a nonlinear system of hyperbolic conservation laws is considered. Diffusion processes can be included as well, but the numerics is not implemented for higher order derivatives, such as third or fourth derivatives as they occur in solitary waves or in the biharmonic equation. Fluxes can always be partitioned in their hyperbolic (finite propagation speed) part and other processes like diffusion, dispersion etc. A transformation is used from physical space to computational space that comprises a set of connected blocks (regular shaped boxes for structured grids) or a set of connected domains (equal size, unstructured grid). The boundaries of neighboring blocks or domains are connected by a set of halo cells of size two, i.e., there is an overlap of two cells between any two neighboring blocks or domains. This is a restriction in the current Physics-Numerics package. The ***JUSTGrid*** allows the selection of any number of halo cells.

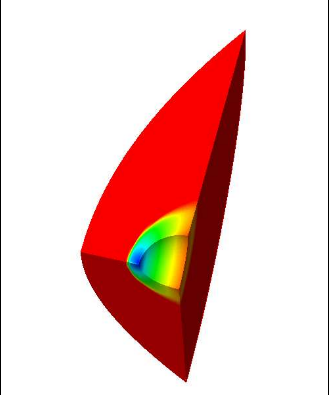
Illustration 7 *3D sphere, 2000 iterations, AoA 0.0, Mach 2.0*

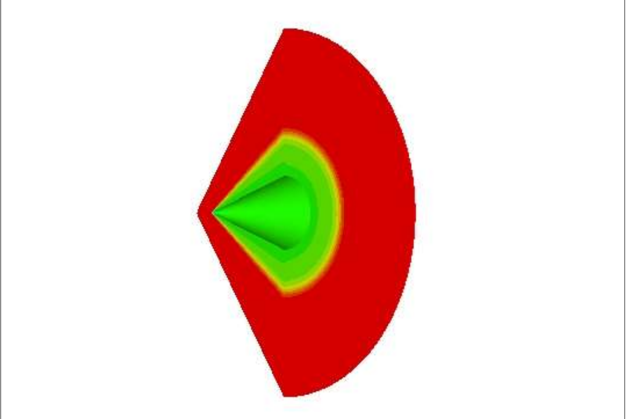
Illustration 8 *3D cone, computed with JUSTSolver, Mach-number distribution, AoA 0, Mach 2.0*

The coordinate free representation in integral form of the system of conservation laws can be written as

$$\frac{\partial}{\partial t}\int_V \boldsymbol{U}\,dV + \oint_{A(V)} \boldsymbol{F}\cdot d\,\boldsymbol{A} = \int_V \boldsymbol{w}\,dV$$

where $\boldsymbol{U}$ denotes the vector of conserved variables and $\boldsymbol{F}$ is the flux tensor.

VI. Java Based Flow Visualization with JUSTVIS

To fulfill the requirements for a platform independent application for CFD, a solution to post-process the produced data should be provided.

Based on this requirement, a pure Java visualization and analysis system, called **JUSTVIS**, is in development using the VisAD framework based on Java3D. With Java3D as the basis of the application, the graphics hardware on all major platforms like Solaris, AIX, Linux, Windows, and also on MacOS X is supported. VisAD offers the basic functionality for a visualization system like data handling for all major types of data (structured and unstructured grids, multiblock grids and hybrid once), color management, user interaction, and the basic visualization techniques like isosurfaces, vector plots, pseudo color mappings, etc.

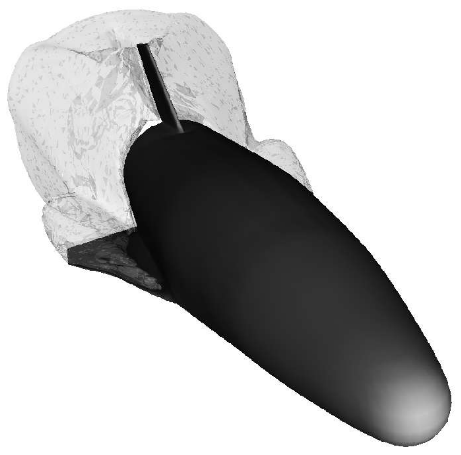
Figure 1: *Visualization of a fully 3D configuration, namely the EXTV (European Experimental Test Vehicle), with the body surface and the isosurface at T = 209K.*

Fig. 1 shows an example of a 3D visualization with an isosurface.

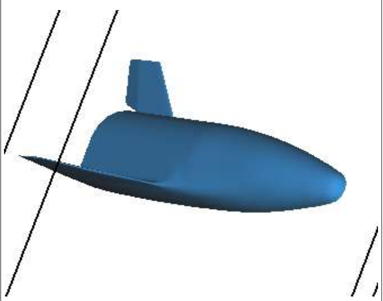
Figure 2 *Surface of the investigated configuration from a computation with a symmetry plane.*

JUSTVIS provides additional analysis methods for, e.g. geometry handling, that helps the user with the analysis process. One example is the fully automatic extraction of surfaces from both full configurations and computations with a symmetry plane. Fig. 1 and Fig. 2 are showing the results for both cases.

To be able to extend other Java written applications with post-processing capabilities **JUSTVIS** will be transformed into a library.

VII. Conclusions

G. JUST

For an existing grid we are now able to provide 100% pure Java based applications for all parts of a simulation for systems of hyperbolic conservation laws, based on the integral form of the conservation equations.

With **JUST** we build a modern, well structured, easy to use and extensible framework (**JUSTGRID**), a sample implementation of a flow solver (**JUSTSOLVER**) and a set of easy to use tools for: post-processing and visualization (**JUSTVIS**), interactive steering (**JUSTCONTOLCENTER** [1]) and online visualization of a running simulation (**JUSTGRXTOOL**[1]).

H. Java High Performance Computing Guidelines

There are two important hints:

1) you have to have an minimum of threads and numerical load before the HotSpot Server VM will show its capabilities. In this case the number of threads is exactly the half number of CPUs. This must be compared with other SMP systems.
2) The system doesn't need additional „expensive" kernel resources if you are running much more threads than installed CPUs in the system. There is a simple rule for multithreaded SMP running environments like these: start a minimum of 4 times more threads than installed CPUs in the system and you get a good dynamic load balancing for free.

I. Java

Today Java has become *THE* modern object oriented language for HPCC, providing excellent performance (at least as fast as C++) and outstanding parallelization features, as well as internet and security features already implemented in the core system.

1. Acknowledgments

This work was partly funded by Arbeitsgruppe Innovative Projekte (AGIP), Ministry of Science, Hanover, Germany under Efre contract

The authors are grateful to Profs. Mark Cross and Mayur Patel, University of Greenwich, London, U.K. for many stimulating discussions.

We are grateful to Mr. Jean Muylaert from ESA, ESTEC for technical information and discussions.

2. References

[1] Ludewig, T., Häuser, J., Gollnick, T., Paap, H.G.: *JUSTGrid A Pure Java HPCC Grid Architecture for Multi-Physics Solvers Using Complex Geometries.* 42th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2004-1091 Reno, NV, USA, 5-8 January 2004.

[2] Fatica, M., Jameson, A., Alonso, J., J.: *StreamFLO: an Euler solver for streaming architectures.* 42th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2004-1090 Reno, NV, USA, 5-8 January 2004.

[3] *Science and technology Shaping the Twenty-First Century*, Executive Office of the President, Office of Science and technology Policy, 1997.

[4] Foster, Ian (ed.): *The Grid: Blueprint for a new Computing Infrastructure*, Morgan Kaufmann Publishers, 1999.

[5] Häuser, J., Ludewig, T., Gollnick, T., Williams, R.D.: *An innovative Software for HPCC.,* ECCOMAS 2001, Computational Fluid Dynamics Conference, Swansea, September 2001, UK

[6] Häuser, J., Ludewig, T., Williams, R.D., Winkelmann R., Gollnick T., Brunett S., Muylaert J.: *A Test Suite for High-Performance Parallel Java,* Advances in Engineering Software, 31 (2000), 687-696, Elsevier.

[7] Ginsberg, M., Häuser, J., Moreira, J.E., Morgan, R., Parsons, J.C., Wielenga, T.J. .: *Future Directions and Challenges for Java Implementations of Numeric-Intensive Industrial Applications*, 31 (2000), 743-751, Elsevier.

[8] Moreira, J.E., S. P. Midkiff, M. Gupta, From Flop to Megaflop: *Java for Technical Computing*, IBM Research Report RC 21166.

[9] Moreira, J.E., S. P. Midkiff, M. Gupta, *A Comparison of Java, C/C++, and Fortran for Numerical Computing*, IBM Research Report RC 21255.

[10] Häuser J., Williams R.D, Spel M., Muylaert J., ParNSS: *An Efficient Parallel Navier-Stokes Solver for Complex Geometries*, AIAA 94-2263, AIAA 25th Fluid Dynamics Conference, Colorado Springs, June 1994.

[11] Häuser, J., Xia, Y., Muylaert, J., Spel, M., *Structured Surface Definition and Grid Generation for Complex Aerospace Configurations*, In: Proceedings of the 13th AIAA Computational Fluid Dynamics Conference -Open Forum, June 29 - July 2, 1997, Part 2, pp. 836-837, ISBN 1-56347-233-3.

[12] Häuser J., Williams R.D., *Strategies for Parallelizing a Navier-Stokes Code on the Intel Touchstone Machines*, Int. Journal for Numerical Methods in Fluids 15,51-58., John Wiley & Sons, June 1992.

[13] Häuser, J., Ludewig, T., Gollnick, T., Winkelmann, R., Williams, R., D., Muylaert, J., Spel, M., *A Pure Java Parallel Flow Solver*, 37th AIAA Aerospace Sciences Meeting and Exhibit, AIAA 99-0549 Reno, NV, USA, 11-14 January 1999.

[14] Winkelmann, R., Häuser J., Williams R.D, *Strategies for Parallel and Numerical Scalability of CFD* Codes, Comp. Meth. Appl. Mech. Engng., NH-Elsevier, 174, 433-456,1999.

[15] Häuser, J., Eiseman, P., Xia, J, Cheng, Z., *Parallel Multiblock Structured Grids, Handbook of Grid Generation*, CRC Press LLC, 12-1, 0-8493-2687-7,1999.