JUSTGRID A Pure Java HPCC Grid Architecture for Multi-Physics Solvers Performance and efficiency results from various Java solvers.

Thorsten Ludewig[1]
University of Applied Sciences Braunschweig/Wolfenbüttel, Computing Center, Wolfenbüttel, GERMANY

Periklis Papadopoulos[2]
San Jose State University, San Jose, CA, UNITED STATES

Jochem Häuser[3], Torsten Gollnick[4] and Wuye Dai[5]
University of Applied Sciences Braunschweig/Wolfenbüttel and HPCC-Space GmbH, Salzgitter, GERMANY

Jean-Marie Muylaert[6]
ESA/ESTEC, Noordwijk, THE NETHERLANDS

and

Hans-Georg Paap[7]
HPC Consultant Barbing, GERMANY

[Keywords] Hierarchical parallel computer architecture, Java multi-physics 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 [2] the Java Ultra Simulator Technology (***JUST***[8]) solver was presented, whose parallel strategy is based on the Java thread concept. While the thread concept works well with an SMP (Symmetric MultiProcessor) architecture, it cannot be applied directly to hybrid parallel systems, comprising nodes with distributed memory as well as multiple processors per node, which are sharing memory. In addition, as was demonstrated using our Java test suite [2, 6, 7], substantial progress has been made over the last three years concerning Java's numerical performance as well as parallel efficiency. The test cases from the test suite (e.g. matrix multiplication, Mandelbrot set or a Laplace solver) were utilized to perform (almost) one-to-one source code comparisons between Java and C++.

In this paper we will present results and performance comparisons for **CFD (Computational Fluid Dynamics)** and also for **MHD (Magneto-Hydro Dynamic)** simulations for 1D and complex 2D as well as 3D geometries,

[1] Central Systems & IT-Division Head, CSIT, CC, UASW, th@uasw.edu, AIAA Member.
[2] Professor, Mechanical & Aerospace Engineering, Perikis.Papadopoulos@sjsu.edu, AIAA Member
[3] Director, HPCC-Space GmbH, jh@hpcc-space.de, Senior member AIAA, member SSE.
[4] Senior Scientist, HPCC-Space GmbH, tg@hpcc-space.de
[5] Senior Scientist, HPCC-Space GmbH, wd@hpcc-space.de
[6] Head, Aerodynamics and Aerothermodynamics Section, ESA/ESTEC, jmuylaer@estec.esa.nl
[7] Senior HPC Consultant, hgp@hpcc-space.de
[8] Name coined by Dr. Jean-Luc Cambier, Senior Research Scientist, AFRL/PRSA, Propulsion Directorate, Edwards AFB, CA, UNITED STATES

using the two components of ***JUST***, namely ***JUSTGrid*** and ***JUSTSolver*** on several different parallel computer architectures. First, the goal is to 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 in UML (Unified Modeling Language) format.

During the last three years there has been enormous change in computing and communications hardware. In the midst of these demands and changes the question arises how to build the simulation software capable of exploiting the new hardware, dealing with complex three-dimensional geometries, running in parallel, being platform (architecture) independent, and being able to access geographically distributed computational resources via the Internet. In addition, the questions of geometric modeling of complex configurations (preprocessing stage) and visualization of computed results arise (post-processing). Visualization and solution feature extraction along with data extraction and compression are of prime importance to deliver the relevant information to the design engineer.

To satisfy the above demands along with the additional requirements of code parallelism, code maintainability and portability, code security, graphics user interfaces (GUI), and data base connectivity to visualize or access data distributed over different computer architectures connected by the Web, requires a completely new approach. With procedural programming languages like Fortran or C or even C++, these goals cannot efficiently be achieved.

Attempts have been made to provide such a computational *Grid* by developing a special computational infrastructure, providing both services and programming tools. With the advent of the Java language in 1996, a general sophisticated object-oriented programming tool is available that provides full coverage of all programming needs on the Internet, and also ensures security. Thus the computational Grid for the Internet can be built entirely in Java in a transparent, object-based approach, termed ***JUSTGrid***. This includes high-performance (parallel) computing as well as data intensive computing utilizing the available computing platforms and network infrastructure.

II. JUSTGrid

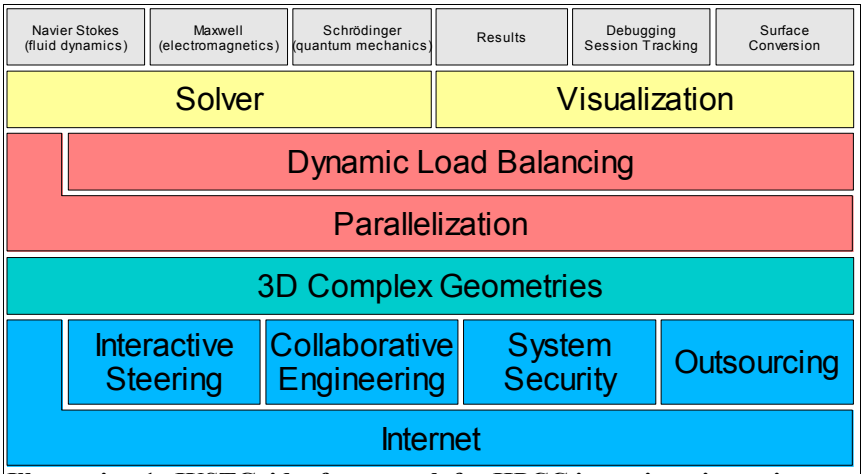

Illustration 1: JUSTGrid a framework for HPCC in engineering, science and life sciences.

JUSTGrid is a completely Java based software environment for the user/developer of HPC (High Performance Computing) software. ***JUSTGrid*** takes care of the difficult task 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, even at runtime.

JUSTSolver is a pure Java CFD solver plug-in for ***JUSTGrid***, based on finite volume technique, and thus can be used for any kind of hyperbolic system of nonlinear partial differential equations formulated in integral form.

III. Results

A.Computational Fluid Dynamics (CFD)

Several simulations were performed to ensure the correct working of the different layers of the ***JUSTGrid***

framework. ***JUSTGrid*** is the core of ***JUST*** the Java Ultra simulator technology. Hence outmost care was taken to prove that ***JUSTGrid*** works absolutely correct. The solvers implemented in package ***JUSTSolver*** will test the ***JUSTGrid*** functionality, performance and efficiency. At the present state numerical and physical accuracy of the scheme and physical validity of the model are of lesser importance. Therefore, in some computations, a Laplace solver was used for a CFD problem, see below.

1. 3D Cone

Simple 3D cone, 8 blocks, 5,832 grid points, 4,096 cells without halo cells

The cone was selected because it is a well known test case. It was thus possible to check nearly the complete functionality of ***JUSTGrid.***

Illustration 2 shows how the ***JUSTGrid*** GRX3D Tool can be used to prepare a simulation run. ***JUSTGrid*** GRX3D is also based on the ***JUSTGrid*** framework and uses the same loaders and utility classes as ***JUSTSolver*** to visualize a grid. It is the very first test to check if ***JUSTGrid*** can handle a given grid. In addition to the visualization one can specify solver specific parameters like „max number of iterations", „Mach number" or „Dt". These parameters are **not** predefined but depend on the selected solver.

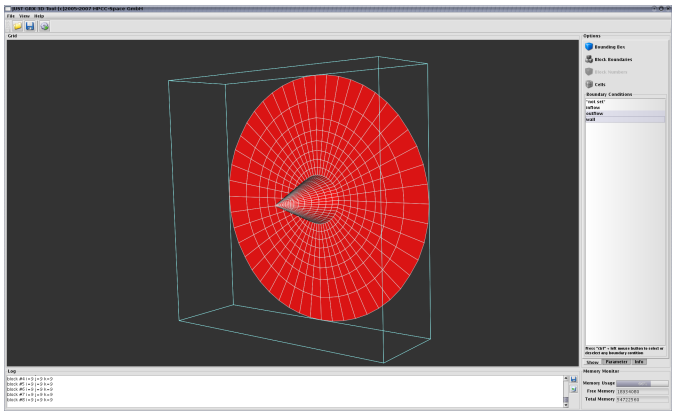

Illustration 2: JUSTGrid GRX3D, simulation preparation tool, showing the grid of the cone wall and the outflow face.

a)***JUSTSolver*** Laplace 3D

The Laplace solver uses Dirichlet boundary conditions, that means in this case inflow (v=1) and wall (v=0) boundaries have fixed values. At the outflow boundary extrapolation is used that means the value will be transported out of the solution domain.

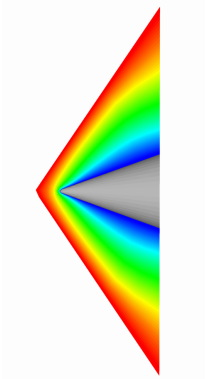

Illustration 3: JUSTSolver Laplace 3D, Cone, showing one slice on the y-plane.

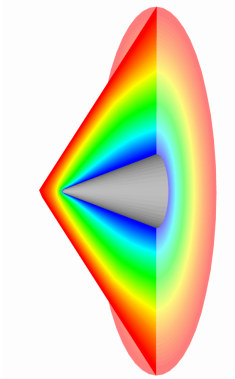

Illustration 4: JUSTSolver Laplace 3D, Cone, showing the outflow boundary.

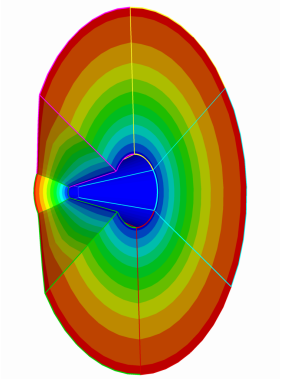

Illustration 5: JUSTSolver Laplace 3D, Cone, showing block edges with one deactivated block.

The following tasks could be verified with this simple test case:

1. Parallelization - ***JUSTGrid*** starts one Thread per block and one monitor thread. All available processors are been used by the simulation. If the computational grid has less blocks than the compute system's number of processors then the surplus processors will be idle.
2. Synchronization - ***JUSTGrid*** implements a loose synchronization between the neighboring blocks. Therefore it is possible that neighbor blocks are one iteration ahead.

3. Communication - ***JUSTGrid*** is also responsible for the boundary update between the neighboring blocks. One can specify any number of halo cells.

b)***JUSTSolver*** Euler3D (1st order, explicit, structured multiblock) compared with CFD++ (2nd order, unstructured)

To test the numerical correctness of ***JUSTSolver*** Euler3D the result of the cone simulation is compared with the result from a commercial CFD solver (CFD++).

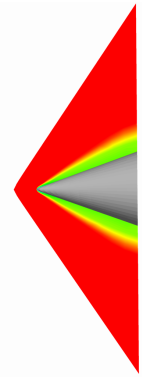

Illustration 6: JUSTSolver Euler 3D showing Mach number solution in symmetry plane.

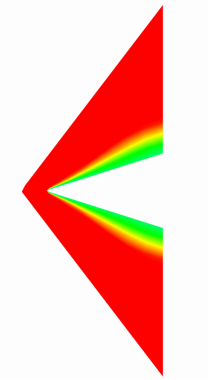

Illustration 7: CFD++ comparison simulation.

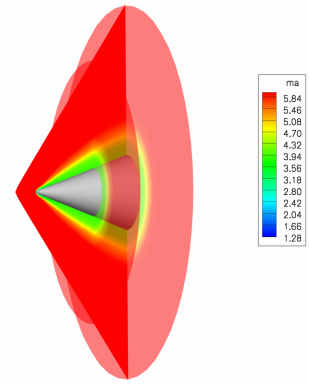

Illustration 8: JUSTSolver Euler 3D with legend and two slices.

Showing the Mach number distribution for a Mach 6.0 simulation after 2000 iterations. While the ***JUSTSolver*** Euler3D simulation is only 1st order accurate, compared with the CFD++ result the Mach number distributing differs not much.

2. European Experimental Test Vehicle (EXTV)

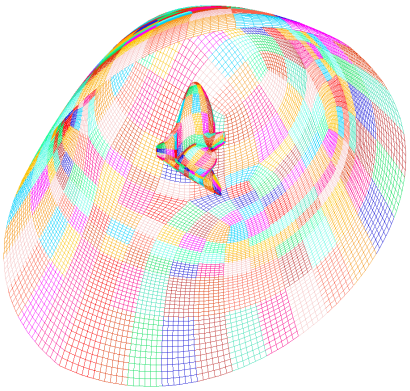

Illustration 9: GridPro™ grid, showing inflow and wall boundaries.

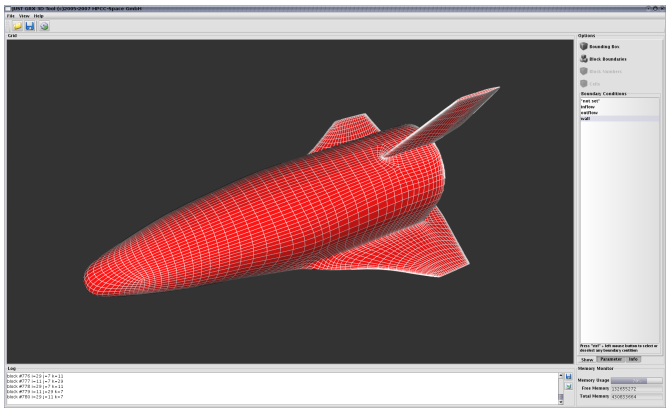

Illustration 10: JUSTGrid GRX3D Tool, showing EXTV wall boundary.

With 780 blocks, 755300 grid points and 538752 cells without halo cells is the EXTV grid a serious test case for larger simulations. The size of this test case is ideal for efficiency and speedup tests. Because it has a reasonable number of blocks, and produces enough numerical work load to achieve a homogeneous dynamic load balancing over all available processors. The simulations were run on a Sun Microsystems *Sun Fire V880* server with 8 UltraSPARC III processors (1.2GHz) and 32GByte main memory running Solaris 10 06/06

c)***JUSTSolver*** Laplace 3D

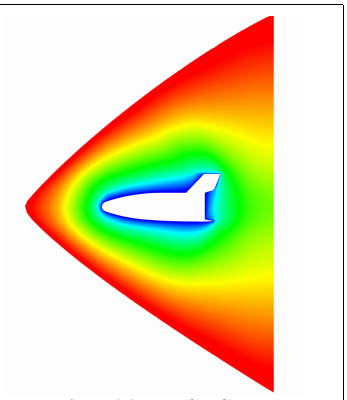

Illustration 11: JUSTSolver Laplace 3D, EXTV showing one slice at the y-plane.

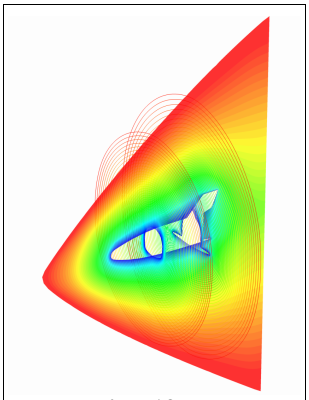

Illustration 12: JUSTSolver Laplace 3D, EXTV 3D view.

The ***JUSTSolver*** Laplace 3D is again used to test parallelization, synchronization and communication features of ***JUSTGrid*** for these large configurations. In order to produce sufficient numerical load the computation was done for 2,000 iterations.

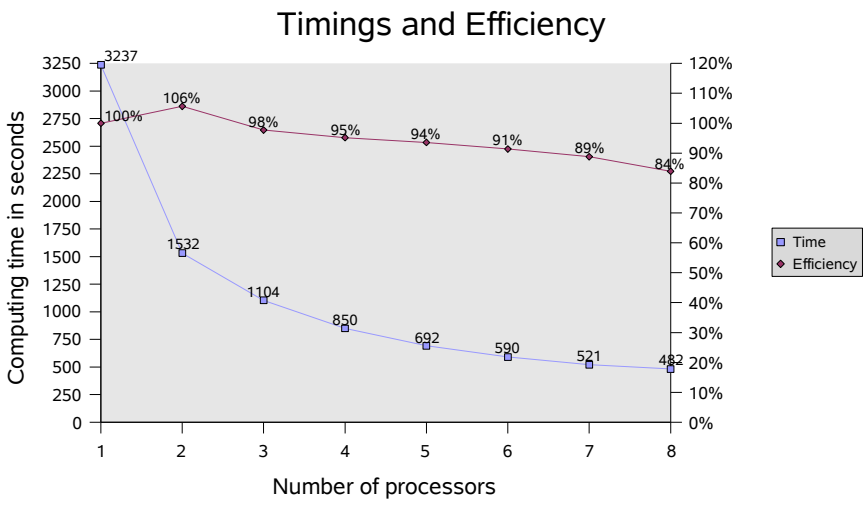

Illustration 13: Timings and efficiency results for 1 to 8 processors, running 2,000 iterations with *JUSTSolver* Laplace 3D on a 780 blocks EXTV grid.

The super linear speedup achieved from one to two processors is observed in many different Java programs. This reflects the behavior of Java's HotSpot compiler. Using only one processor the profiling task of the HotSpot compiler itself consumes appreciable time to find the program's most time consuming regions (hot spots).

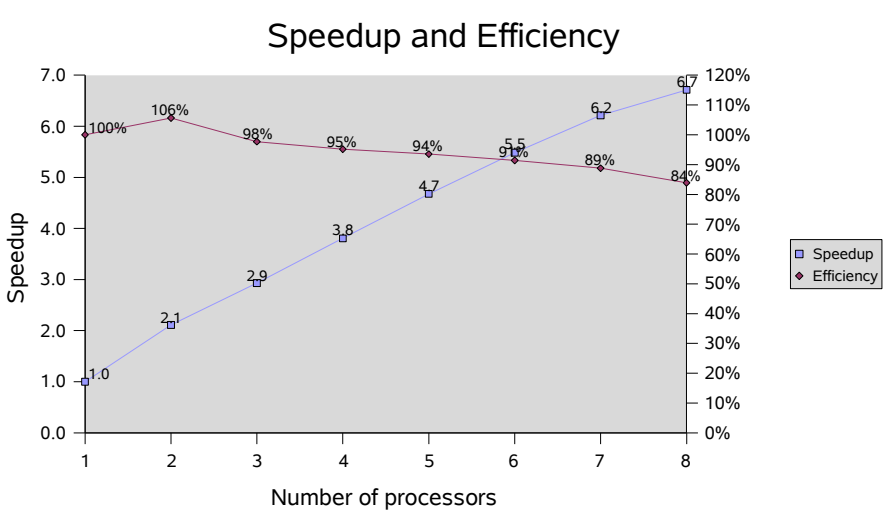

Illustration 14: Speedup and efficiency results for 1 to 8 processors, running 2000 iterations with *JUSTSOLVER* Laplace 3D on a 780 blocks EXTV grid.

JUSTSOLVER Laplace 3D demonstrates excellent (almost linear) speedup at the hardware configuration utilized.

d)***JUSTSOLVER*** Euler3D

To compare the Java based ***JUSTSOLVER*** Euler3D with a flow solver written in 'C' (ParNSS) an EXTV simulation using a free stream value of Mach 8.0 and an angle of attack 0.0 was chosen.

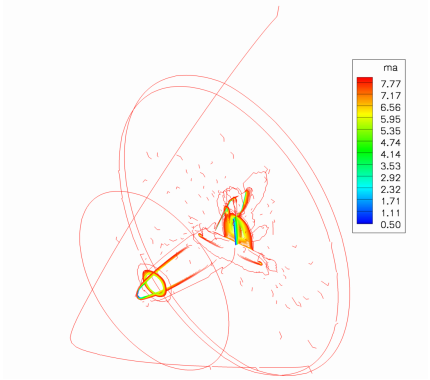

Illustration 15: JUSTSolver Euler 3D, EXTV, Mach number distribution.

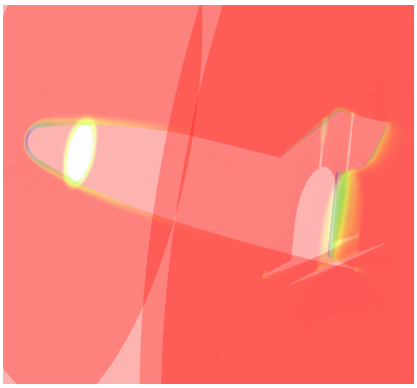

Illustration 16: JUSTSolver Euler 3D, EXTV, Mach number distribution.

ParNSS is a legacy 3D structured multiblock code written in 'C'. ParNSS is utilizing the MPI (Message Passing Interface) library to implement the parallelization and communication between neighboring blocks. The implemented numerics for the flux computation (van Leer) for ParNSS and ***JUSTSOLVER*** Euler3D are almost identical (99%) at the source code level. Hence, this provides an excellent opportunity to perform reliable performance comparisons between a 'C' based CFD solver and a Java based flow solver.

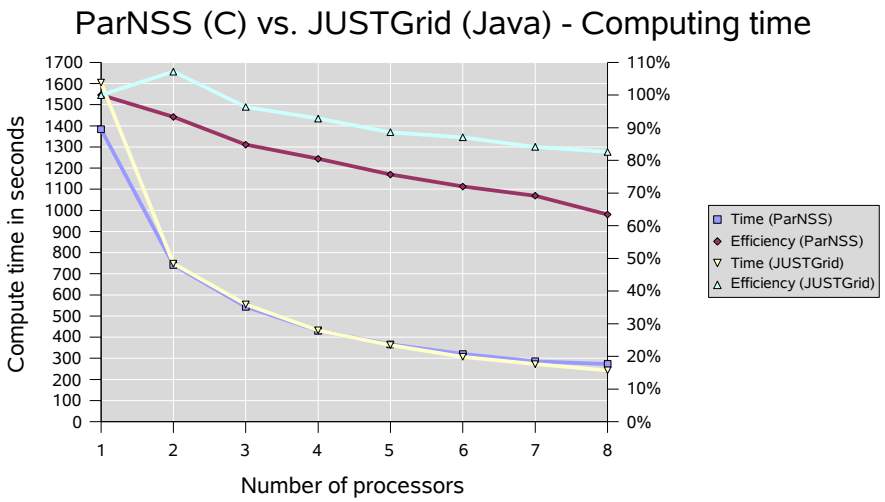

Illustration 17: Computing time an efficiency results for 1 to 8 processors, running 200 iterations with JUSTSolver Euler 3D and ParNSS on a 780 blocks, 755,300 grid points, 538,752 cells EXTV grid.

Due to the profiling task of the HotSpot compiler the Java solver is much slower with one processor than the 'C' solver. Employing 5 or more processors ***JUSTSolver*** Euler3D is faster than ParNSS. For 8 processors the time difference is already more than 30 seconds for only 200 iterations.

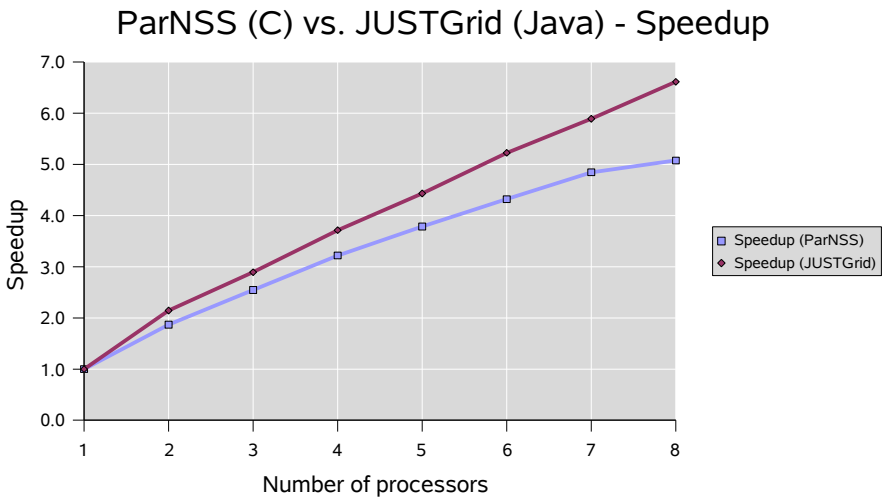

Illustration 18: Speedup results for 1 to 8 processors, running 200 iterations with JUSTSolver Euler 3D and ParNSS on a 780 blocks, 755,300 grid points, 538,752 cells EXTV grid.

JUSTGrid / JUSTSolver Euler3D achieves better linear speedup than ParNSS. In [6] it is shown that Java programs can achieve linear speedup for numerical applications on large SMP machines with more than 20 processors.

B. Performance progress of the Java Virtual Machine

In order to demonstrate the performance progress three different versions of the JVM are compared in Tables 1 and 2 for two different hardware systems. Running 100 iterations on a 780 blocks EXTV grid using ***JUSTSOLVER*** Euler 3D to demonstrate the performance progress of the Java Virtual Machine (JVM) on different architectures.

a)SPARC processor architecture

JVM Version	32 Bit	64 Bit
1.4.0_12 server	172.57	332.83
1.5.0_10 server	136.68	203.12
1.6.0_fcs server	137.92	159.34

Table 1: JVM performance progress on a Sun Microsystems V880 with 8 processors (UltraSPARC III, 1.2GHz), 32 GB main memory running Solaris 10 06/06. Execution times are in seconds.

On the SPARC architecture the 64 Bit version of the JVM produces the best performance progress. The 1.6.0 version of the JVM just has been released, and next sub-releases definitely will be faster. This behavior was observed in all former „first customer shipment" (FCS) JVM releases (e.g. 1.4.0_fcs, 1.5.0_fcs).

b)AMD Opteron processor architecture

JVM Version	32 Bit	64 Bit
1.4.0_12 server	793.21	
1.5.0_10 client	211.12	
1.5.0_10 server	116.25	155.24
1.6.0_fcs server	126.31	156.26

Table 2: JVM performance progress on a Sun Microsystems U40 with 2 Dual Core processors (AMD Opteron 280), 8 GB main memory running Solaris 10 06/06.

The highest performance boost was achieved from JVM 1.4 to JVM 1.5. Similar to the SPARC architecture, the JVM 1.6.0 (FCS) is slightly slower than the actual JVM 1.5. But the JVM 1.6 is about 20% faster during the loading and writing of a grid.

C. Magneto Hydro Dynamic (MHD)

1. *Brio-Wu's Shock-Tube*

The solution was computed up to time t = 0.25s, because the numerical solution has reached the end of the computational domain. Computational results show excellent agreement with the original results. This shows that the physics and numerics are implemented correctly.

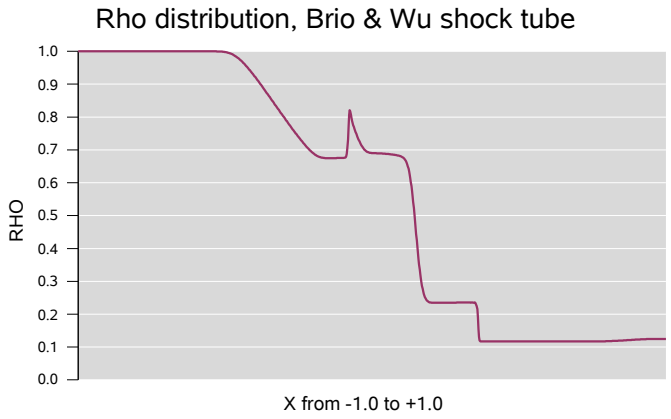

Illustration 19: 1D MHD solution, rho (density) distribution for well known Brio & Wu shock tube.

2. *2D Riemann-MHD*

The density distribution obtained for a Riemann-MHD computation is shown in Illustration 20. The solution domain is given by a rectangle [-0.3,0.3;-0.3,0.3]. The initial conditions are

initial data ($\boldsymbol{B}_0 = \frac{1}{\sqrt{2}}(1,0,0)^T$)				
	$\rho_0(x,y)$	u_x	u_y	$p_0(x,y)$
x<0, y<0	10	0	0	15
otherwise	1	0	0	0.5

The original experiment can be found in M. Torrilhon, *Zur Numerik der idealen Magnetohydrodynamik*, ETH Thesis Nr. 15353, Eidgenössische Technische Hochschule, Seminar für angewandte Mathematik (2003,Nov.), pp. 150-154.

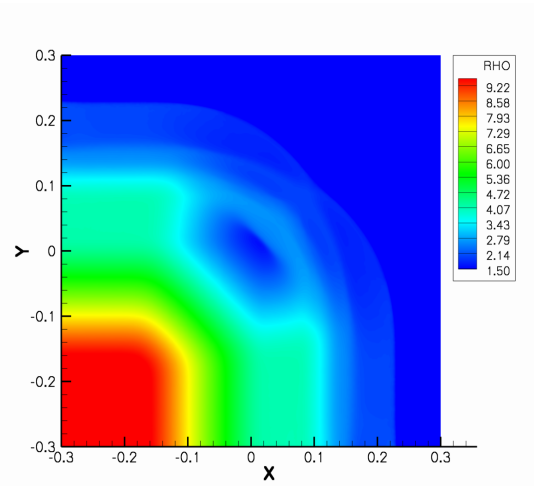

Illustration 20: 2D MHD solution, rho (density) distribution for a Riemann solver.

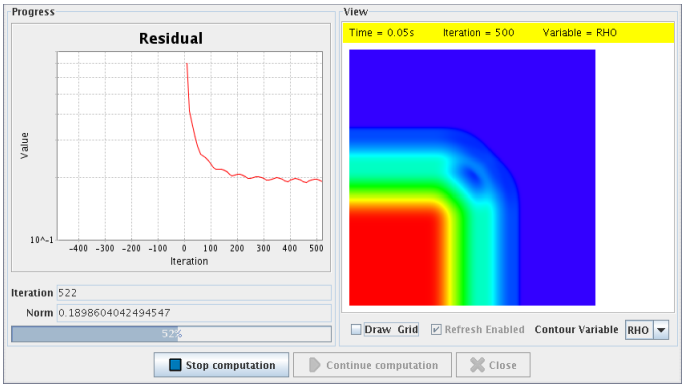

Illustration 21: JUSTGrid GRXMonoblock Tool GUI, showing online visualization while the 2D MHD Riemann solver is running.

3. ***JUSTGRID's*** *GRXMonoblock Tool*

Illustration 21 shows the online visualization feature of ***JUSTGRID***'s GRXMonoblock Tool. It gives a good impression about the current state of the simulation. Clearly, it is not meant to be a replacement for visualization tools like TecPlot™ or Ensight™. Another useful feature of ***JUSTGRID***'s GRXMonoblock Tool is the QuickTime™ movie generation during the simulation.

IV. Acknowledgements

The work presented here is part of the Ph.D. thesis of the first author submitted to Greenwich University, London, U.K.

This work was partly funded by Arbeitsgruppe Innovative Projekte (AGIP) and EFRE, Ministry of Science and Culture (MWK), Hannover, Germany.

This research was also partly performed by the Air Force Office of Scientific Research (London, U,K.), Air Force Material Command, USAF under grant number FA8655-07-1-3027. We are grateful to Dr. Jean-Luc Cambier, Propulsion Directorate, Edwards Air Force, Base for numerous helpful comments concerning the MHD calculations.

A major part of this work was performed under ESA's Let-SME program, contract number NL 18732.

The Government of Lower Saxony, Germany, the European Commission, the European Space Agency, and the U.S. Government are authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright notation thereon.

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

V. References

[1] Ludewig, T., Häuser, J., Gollnick, T., Paap, H.G.: A Java Based High Performance Solver for Hierarchical Parallel Computer Architectures. 43rd AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2005-1383 Reno, NV, USA, 10-13 January 2005.

[2] 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.

[3] 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.

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

[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., 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.

[13] 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.