

CONTRACT NUMBER 508830

DEISA
**DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR
 SUPERCOMPUTING APPLICATIONS**

European Community Sixth Framework Programme
 RESEARCH INFRASTRUCTURES
 Integrated Infrastructure Initiative

JRA3: Final report on EUTERPE code

Deliverable ID: DEISA-D-JRA3-6

Due date: April 30, 2007
Actual delivery date: May 28, 2007
Lead contractor for this deliverable: RZG, Germany

Project start date: May 1st, 2004
Duration: 4 years

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	X
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Contents

Table of Contents.....	1
1 INTRODUCTION	2
1.1 Executive Summary.....	2
1.2 References and Applicable Documents	2
1.3 List of Acronyms and Abbreviations	3
2 COMPUTATION OF TURBULENCE IN FUSION EXPERIMENTS IN EUROPE.....	4
3 FINAL REPORT ON THE EUTERPE CODE.....	4
3.1 Optimizing EUTERPE for large sparse matrices	4
3.2 Enabling EUTERPE for a matrix free solver.....	6
4 ENABLING OF THE GEM (GYROFLUID ELECTROMAGNETIC) CODE.....	6
4.1 Introduction.....	6
4.2 Single processor performance optimization	8
4.3 Scaling properties of GEM.....	9
4.4 Concept for enhancement of scalability.....	11
5 PROOF OF VERY HIGH SCALABILITY OF ORB5.....	12
6 PROOF OF VERY HIGH SCALABILITY OF GENE.....	14
7 FURTHER ROADMAP.....	14

1 Introduction

1.1 Executive Summary

This document “JRA3: Final report on EUTERPE” is the 36 month deliverable DEISA-D-JRA3-6 for Joint Research Activities in Plasma Physics.

A final report on the EUTEPRE code is presented. In addition to the enabling work already done in the framework of DEISA-D-JRA3-5, the code was optimized for large sparse matrices in several ways: the matrix construction time has been decreased by an order of magnitude; all the processors can now take part in the matrix solve of the PETSc library; support has been given for the development of a very efficient parallel matrix free solver.

As planned and announced in DEISA-D-JRA3-5, the work on the new code candidate for enabling, the GEM code, has been started and is under way. It was analysed in detail and bottlenecks were identified. Performance enhancements were done at several levels. The single processor performance was increased by more than 50 %. The investigation of the original parallelization scheme along the magnetic field line coordinate triggered a laborious index structure change still in progress. Further, a working plan for the scheduled x-direction parallelization has been prepared.

In addition, the ORB5 code was ported and tested for very high scalability on the BlueGene/L system at IBM Watson Research Centre. Good scalability was demonstrated with a parallel efficiency of 88 % on 8192 processors (relative to 1024 processors) for a fixed problem size requiring about 400 GB of main memory. The corresponding weak scaling from 1024 to 8192 processors revealed an even better parallel efficiency of 100%.

In a joint effort with eSA4-T1 (eDEISA), GENE was also even further parallelized and also ported to BlueGene/L. From 1024 to 16384 processors, GENE achieved a parallel efficiency of 89% in strong scaling mode. In weak scaling mode, GENE achieved a parallel efficiency of 99% on 32678 processors (relative to 2048 processors).

As next code candidate for enabling, the electromagnetic particle-in-cell (PIC) code GYGLES has been selected upon request.

1.2 References and Applicable Documents

[1] C.C. Kim and S.E. Parker: Massively Parallel Three-Dimensional Toroidal Gyrokinetic Flux-Tube Turbulence Simulation. *J. Comp. Phys.*, **161**: 589, 2000

[2] R. Hatzky: Domain Cloning for a Particle-in-Cell (PIC) Code on a Cluster of Symmetric-Multiprocessor (SMP) Computers, *Parallel Computing*, **32**: 325, 2006

[3] B. Scott: Free-energy conservation in local gyrofluid models. *Phys. Plasmas*, **12**: Art. No. 102307, 2005

[4] A. Kendl and B. Scott: Magnetic shear damping of dissipative drift wave turbulence, *Phys. Rev. Lett.*, **90**, Art. No. 035006 , 2003

[5] R. Kleiber and B. Scott: Fluid simulations of edge turbulence for stellarators and axisymmetric configurations. Phys. Plasmas, **12**, Art. No. 102507, 2005

[6] G.E. Karniadakis, M. Israeli and S.A. Orszag: High-order splitting methods for the incompressible Navier Stokes equations. J. Comput. Phys., **97**: 414, 1991

[7] A. Arakawa: Computational design of long-term numerical integration of the equations of fluid motion: Two-dimensional incompressible flow. Part I. J. Comput. Phys., **1**: 119, 1966

[8] V. Naulin and A. Nielsen: Accuracy of spectral and finite difference schemes in 2D advection modes. SIAM J. Sci. Comput., **25**: 104, 2003

[9] M. Fivaz, et al.: Finite element approach to global gyrokinetic particle-in-cell simulations using magnetic coordinate, Comp. Phys. Commun., **111**: 27, 1998

1.3 List of Acronyms and Abbreviations

CFN	Centro de Fusão Nuclear, see http://www.cfn.ist.utl.pt/
CRPP	Centre de Recherches en Physiques des Plasmas, Lausanne see http://crppwww.epfl.ch
DECI	DEISA Extreme Computing Initiative
ESSL	IBM Engineering and Scientific Software Library, see http://www-03.ibm.com/systems/p/software/essl.html
FFTW	Fastest Fourier Transform in the West, see http://www.fftw.org
Fortran	FORmula TRANslation, scientific programming language
FZ-Jülich	Forschungszentrum Jülich, see http://www.fz-juelich.de
IPP	Max-Planck-Institut für Plasmaphysik, Garching, see http://www.ipp.mpg.de
ITER	International Thermo-Nuclear Experimental Reactor, see http://www.iter.org
JRA	Joint Research Activity
MKL	Intel Math Kernel Library, see http://www.intel.com/cd/software/products/asmona/eng/perflib/mkl/index.htm
MPI	Message-Passing Interface, see http://www.mpi-forum.org
PETSc	Portable, Extensible Toolkit for Scientific Computation, see http://www-unix.mcs.anl.gov/petsc/petsc-as/
PIC	Particle-In-Cell
RZG	Rechenzentrum Garching, see http://www.rzg.mpg.de
SMP	Symmetric MultiProcessor
UKAEA Culham	the United Kingdom Atomic Energy Authority, Fusion power, see http://www.fusion.org.uk/
Universität Innsbruck	Plasma and Energy Physics, see http://www.uibk.ac.at/th-physik/pep/
WSMP	IBM Watson Sparse Matrix Package, see http://www-users.cs.umn.edu/~agupta/wsmp.html

2 Computation of Turbulence in Fusion Experiments in Europe

There are several outstanding questions regarding the performance of fusion experiments and hence the availability of fusion power in the future. Many are materials and hardware related, but one very important question is related to plasma physics. This is the scaling and character of turbulence and energy transport, including such phenomenology as flows and transport barriers inside the plasma. The results have a direct effect on judgements of how much energetic input is needed to maintain a thermonuclear plasma and even, in certain scenarios, whether a very large reactor class fusion plasma can function at all.

Decades of modelling have found that simplified or statistical treatments are inadequate and direct numerical simulation is necessary, at least within the time scales over which the turbulent state governing transport is established. Both the edge and core regions of the plasma (usually treated as separate problems) are involved, and a major theme of the next stage of research will be to combine them. The necessary mathematical progress is underway and the model equations have recently become available. In terms of computations, large scale models with simplified underlying equations, or smaller models with equations closer to the necessary degree of completion, have been developed.

In order to capture the wide range of scales involved, the computations must be done at very high resolution. The complexity of plasma physics equations make this more difficult than for conventional fluid research. The nature of a sheared and curved magnetic field adds enormously to this. At present day, global models must neglect the dynamics of the electrons, and fluid models are the only ones which can treat the self consistent changes in the background equilibrium. The best techniques are most often split among the approaches: the models which can treat both core and edge do not exist yet. One of the themes in the next stage of progress is that many of these separate approaches should be combined. One such effort is described in Sec. 4.1.

3 Final report on the EUTERPE code

3.1 Optimizing EUTERPE for large sparse matrices

As stated already in the deliverable DEISA-D-JRA3-5, there was an open request from the developing scientist to optimize the (strong) scaling behaviour of EUTERPE for the matrix solution part. In the former enabling work the domain cloning concept [1, 2] had the direct consequence that the solver of the matrix was only parallelized over each clone. Hence, the matrix was solved redundantly on each clone. In general, the parallelization of the matrix solve can be chosen completely independent for the domain cloning concept. As a result, the MPI communicators defined for the domain cloning concept can no longer be used for the parallelized matrix solver. Instead, a new MPI communicator for the PETSc library has been defined which parallelizes the matrix also over each subdomain clone (see section 3.3.3, DEISA-D-JRA3-5). In principle, all processors can now take part in the solution of the matrix problem. However, the construction of the matrix is still done separately from the main program in an initial step and is parallelized on a number of processors which is consistent with the number of

processors on a clone. Subsequently, the main program reads in the corresponding matrix data and distributes them among the clones.

Fig. 1 shows on the left the strong scaling of the new EUTERPE code version for 2^{28} particles with a grid of $64 \times 128 \times 128$ quadratic B-splines (circles) and cubic B-splines (stars) respectively on an IBM p690 node with POWER4 1.3 GHz processors at RZG. The speed-up is quite close to an optimal linear scaling (solid line) and reaches an efficiency of 88 % for quadratic and 90 % for cubic B-splines. However, the speed-up of the PETSc solver routine shown in Fig. 1 on the right is quite poor and contributes significantly to the part of the code which does not scale optimal. For quadratic B-splines hardly any speed-up can be detected while the cubic B-spline discretization shows at least some improvement in the speed-up. The transition from quadratic to cubic B-spline discretization leads to a less sparse matrix structure which gives each processor more compute work compared to the communication overhead.

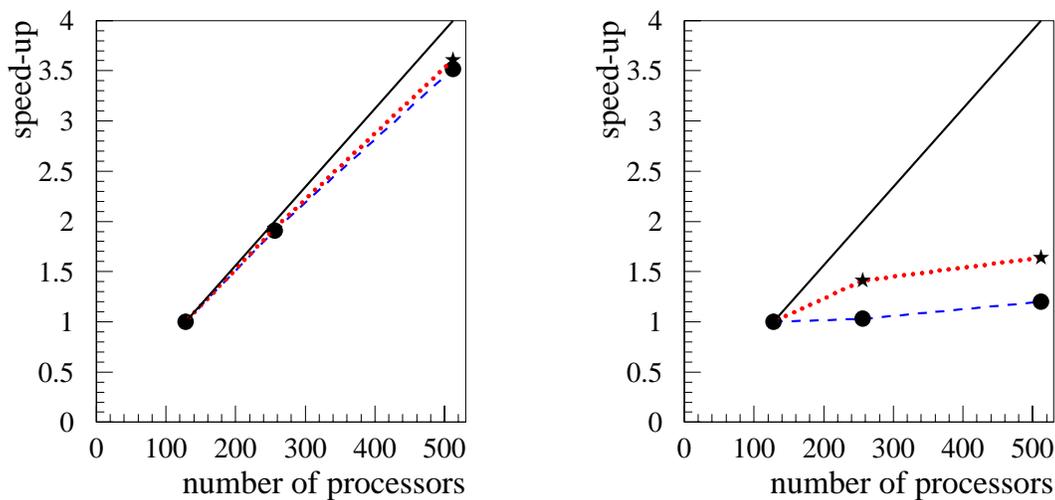


Fig. 1 left: Strong scaling of the new EUTERPE code version for a grid of $64 \times 128 \times 128$ quadratic B-splines (circles) and cubic B-splines (stars) on an IBM p690 with POWER4 1.3 GHz processors at RZG; right: Strong scaling of just the PETSc solver routine; the solid line depicts linear scaling

The results clearly show that the matrix size for the chosen grid becomes too small to be parallelized efficiently on such a large number of processors. Thus, the grid size has to be increased to produce larger matrices which can be solved more efficiently on the number of processors made available by the new MPI communicator. In the framework of the nonlinear simulation proposed for the DECI gyro3d proposal, the grid size has to be increased anyway to at least $64 \times 256 \times 256$. Hence, the nonlinear DECI simulation will clearly benefit from the new MPI communicator for the PETSc library.

The performance of the matrix construction has not been an issue so far as it was quite fast compared to the main program execution time. However, this will change for very large matrices. The main code part of the matrix construction consisted of a fourteen-fold nested DO loop. Hence, even small performance changes in the innermost loops can have drastic impact on the overall performance. It was possible to significantly reduce the matrix construction time by an order of magnitude and has been achieved by pre-computing values and storing them in look-up tables and by reducing the number of nested loops to thirteen.

For validating the extensive usage of PETSc library routines in EUTERPE the WSMP (Watson Sparse Matrix Package) library has also been implemented. If possible it is always useful to cross-check the results of one numerical library with the results of another one especially if the numerical methods differ. So in our case, we could check the results of the iterative PETSc solvers against the results of the direct sparse solver of WSMP based on the Cholesky decomposition method. On the aspect of performance of solving matrices derived from three-dimensional problems, it has been confirmed that for larger matrices the direct solver has too large a fill-in factor to be competitive with an iterative solver.

3.2 Enabling EUTERPE for a matrix free solver

For nonlinear simulation of ITG modes the equation for the adiabatic response of the electrons has to be changed in such a way that the flux surface average of the electrostatic potential has to be subtracted. After discretization with B-splines one arrives at a matrix equation where one part can be stored consistently to the previous section in sparse matrix format. In contrast, the non-local character of the averaging operation nearly completely fills the matrix structure and, for a three-dimensional case, would lead to a matrix which would be too large to be stored even on distributed memory machines. However, the PETSc library gives the possibility to let the user define its own matrix-residual multiplication procedure (matrix shell). With such a routine it is possible to perform the matrix-residual multiplication of the averaging operation without actually having the matrix elements pre-computed and stored. Instead, the matrix elements are calculated when needed. The performance of such a matrix free scheme depends significantly on the efficiency of the construction of the matrix elements. In our case, these matrix elements cannot be constructed locally as they reflect the effect of an averaging procedure over two spatial dimensions, of which one is the direction of parallelization. Hence, the construction of the matrix elements makes it necessary to communicate data among the processors. However, communication is costly compared to computation so it becomes crucial that the algorithm uses as little communication as possible. The work of the developing scientist, Ralf Kleiber, has been supported in such a way that the overhead of the construction of the matrix elements takes only 30 % of the overall matrix-residual multiplication routine deployed by the PETSc library. To keep the priority of the development of this new scheme details will not be shown here.

The EUTERPE code is now in a state to construct and solve large matrices in an efficient manner. Thus, as envisaged in the DECI proposal Gyro3D, the nonlinear three-dimensional ITG simulation seems to be within reach.

4 Enabling of the GEM (Gyrofluid ElectroMagnetic) Code

4.1 Introduction

The GEM code solves nonlinear gyrofluid equations for electrons and one or more ion species in tokamak geometry [3]. Extensions to stellarator cases have been made [4, 5], though the global version is focused on tokamaks. The GEM code has a central role in collaborations involving IPP/Garching, CFN/Lisboa, the University of Innsbruck and FZ-Jülich.

The gyrofluid model equations are a direct low-order moment model of the gyrokinetic equation for each species and make possible computations on very large systems not otherwise possible with present-day or near-term resources. At present, it is the only code capable of simulating electromagnetic turbulence in a self consistent equilibrium, treating both electron and ion temperatures outside the collision dominated fluid regime, valid on scales below the ion gyroradius, and able to treat the entire range of scales from the global to the electron skin depth at realistic parameters. This typically requires a resolution circa 1000 grid points in the perpendicular dimensions for the largest current tokamaks, which will have to be made a factor of 3 or 4 larger to treat the planned ITER experiment. The computational runs must be carried out for rather long times, as the time scale for statistical equilibrium in the large scale “zonal flows” which regulate the turbulence is about 1000 times longer than the lifetime of individual turbulent eddies.

Such large grids with GEM have already been enabled by application of mathematical techniques to the grid on coordinates continuously aligned to the background magnetic field. Magnetised plasma turbulence is strongly anisotropic, such that while a wide scale range persists across the magnetic field, only a limited range of wavenumbers are excited along it. Hence, the mathematical transformations allow the necessary resolution in the third (parallel) coordinate to decrease by a factor exceeding 100 for medium to large systems. This technique represents the state of the art but present-day global codes mostly do not exploit it. Consequently, even with large supercomputer resources (ca. 1024 processors), they cannot approach the necessary perpendicular resolution because that third coordinate also requires high resolution, as on a non-aligned grid the gradients in two coordinates must largely cancel to produce a parallel gradient. This is the reason GEM is usually run with 16 grid points in the parallel direction and nearly never with more than 32. This factor of roughly 100 is saved by mathematics even before coding begins.

Besides the advances achieved by applying subtle mathematical techniques to the equations, the numerical scheme is set up to efficiently exploit the nature of the resulting dynamics. The time step method is third order while requiring only one evaluation of the terms per time step [6], and the spatial discretization preserves the Poisson-bracket properties of the nonlinear operations to hardware accuracy [7]. The combination is stable for both waves and coherent vortices, preserving their overall properties (Hamiltonian conservation, low dispersion) to high accuracy [8]. Standard methods (upwind, Fourier, implicit) are constructed with no input from these properties and typically violate them, usually also at greater computational expense. The GEM code represents the first wholesale import of all the above advances into a single global computation of magnetised plasma dynamics. Hence, when numerical performance is assisted, it represents improvement of a method whose underlying properties are already optimal.

The parallelization strategy consists of domain decomposition in the direction along the magnetic field and has been implemented by using MPI. Hence, the processor count is limited to the grid point count in the coarsest dimension. It is our goal to extend this to radial/parallel decomposition with an additional radial direction of parallelization. Currently available grids for local cases are in the 512 x 256 x 16 or 256 x 256 x 32 ranges. The first two numbers in each case will have to be at least 1000 as noted above as soon as becomes feasible. At the moment, the field solvers use an FFT decomposition to reduce the problem to a tridiagonal matrix solve. The next generation of this code (GEMX) will be fully “nonlocal” in the sense that the matrices to be solved depend on the dynamical variables which change each time step. For this, the Fourier method cannot be used any longer. Instead, the resulting matrices have to be solved in parallel by a sparse matrix solver such as the Watson Sparse Matrix Package (WSMP) or the PETSc library.

Thus, only a scaling of the code to several hundreds of processors will make it feasible to run global cases with the GEMX version which will result in a better understanding of the global phenomena in fusion devices.

4.2 Single processor performance optimization

The enabling work on the GEM code for DEISA started with a scan for potential improvements of the single processor performance. For identification of the most CPU time consuming routines the code has been instrumented by the simple but efficient perf library. The perf library has been programmed by the RZG scientist Reinhard Tisma and gives information about the time spent and the Mflop rate achieved in a detected region (see e.g. Table 1). The regions have to be defined by hand and can be marked by arbitrary labels.

Subroutine	#calls	Time(s)	Inclusive		Time(s)	Exclusive	
			%	MFlops		%	MFlops
main	1	128.141	100.0	<u>356.761</u>	0.008	0.0	0.154
yshift	8000	26.588	20.7	70.239	26.588	20.7	70.239
arakawa	250	18.047	14.1	942.988	18.047	14.1	942.988
upwind2	250	9.392	7.3	143.809	9.392	7.3	143.809
diss	250	10.475	8.2	345.450	10.475	8.2	345.450
visc	250	20.785	16.2	528.002	20.785	16.2	528.002
advance	250	11.969	9.3	209.487	11.969	9.3	209.487
polarise	250	20.709	16.2	345.747	20.709	16.2	345.747
snap_wr	5	0.713	0.6	0.000	0.713	0.6	0.000

Table 1 Performance output of the original GEM code for a single processor run (IBM POWER4 1.3 GHz) of a small test case

Tab. 1 shows the performance output of the original GEM code which is written in Fortran 95 for a single processor run (IBM POWER4 1.3 GHz) of a small test case with an overall Mflop rate of the main loop of 357 Mflops. Further the Mflop rates of the most time consuming regions of the main loop have been listed. In the following, we list the most effective performance changes which result in a performance increase of a factor of 50 % to 538 Mflops in the main loop (see Table 2):

- In the routine `yshift` a circular shift is done in the y-coordinate direction due to a coordinate transformation. The performance has been improved by simplifying the routine and replacing parts of the routine by the intrinsic Fortran 90 routine `CSHIFT` and reducing the number of calls to `CSHIFT` from two to one.
- In the code parts `upwind2` and `visc` temporary arrays for intermediate results were used. Instead the mathematical operations are done now on just one array successively.
- In the code part `advance` different arrays had been copied each time step in a shifted manner due to updating to the new time step. So e.g. for three arrays `a`, `b` and `c` the array `b` is copied into array `c` and `a` into `b`. As result array `a` is vacant to store the data for the next time step and the arrays `b` and `c` store the data of the previous time steps. Instead of copying the data from one array into the other it is much more efficient to allocate all three arrays into one array and just move the last index distinguishing between the former arrays `a`, `b` and `c`. So at the beginning the values 1, 2 and 3 of the last index will map to the former arrays `a`, `b` and `c`. Then a circular shift to the left is done on the values which results in 2, 3

- and 1. So in the framework of Fortran 77 syntax the last index is used as a pointer which can be easily moved for each time step.
- d) The code part `polarise` contains the field solves which are done by an FFT decomposition. Unfortunately, these FFT routines were hard coded and did not have a very high performance. Already in DEISA-D-JRA3-4 a module had been provided for complex one-dimensional FFTs which contained different interfaces to specialized FFT libraries such as the FFTW v3.1 (Fastest Fourier Transform in the West), the IBM ESSL (Engineering and Scientific Software Library) and the Intel MKL (Math Kernel Library). For usage in the GEM code the module has been extended for all three libraries to real to complex and complex to real FFTs in one- and two-dimensions. A significant performance increase could be achieved.

In addition, the GEM code has been checked by the runtime diagnostics of the IBM xlf, Lahey/Fujitsu and Intel Fortran compilers for e.g. undefined variables, substring and array subscripts out of range and not matching procedure arguments. For extended Fortran support the MPI module defined in the MPI-2 standard has been used to check via MPICH the interfaces of all used MPI routines on correctness.

Subroutine	#calls	Time(s)	Inclusive		Time(s)	Exclusive	
			%	MFlops		%	MFlops
main	1	83.103	100.0	<u>537.515</u>	0.004	0.0	2.482
yshift	8000	6.681	8.0	280.614	6.681	8.0	280.614
arakawa	250	17.214	20.7	1014.659	17.214	20.7	1014.659
upwind2	250	6.779	8.2	197.257	6.779	8.2	197.257
diss	250	10.405	12.5	348.035	10.405	12.5	348.035
visc	250	18.204	21.9	602.514	18.204	21.9	602.514
advance	250	5.747	6.9	436.808	5.747	6.9	436.808
polarise	250	7.672	9.2	724.677	7.672	9.2	724.677
snap_wr	5	2.110	2.5	0.000	2.110	2.5	0.000

Table 2 Performance output of the new GEM code for a single processor run (IBM POWER4 1.3 GHz) of a small test case

4.3 Scaling properties of GEM

The original GEM code version has implemented a domain decomposition concept over the spatial coordinate s along the magnetic field lines. The typical grid resolution in the s direction is 16 and in some particular cases 32. Hence, the number of processors used for parallelization is restricted to 32 which is quite a small number when compared to today's supercomputers such as IBM BlueGene/L consisting of ten thousands of processors.

In Fig. 2 the results of a strong scaling of the GEM code on an IBM p690 SMP node with POWER4 1.3 GHz processors at RZG are shown. The test cases consist of two grids of $32 \times 128 \times 16$ (left) and of $32 \times 128 \times 32$ (right) grid points with the coordinates x direction, y direction and s coordinate along the magnetic field. The results of the new version (dashed lines) show a maximal speedup of 13 on 16 processors (left) and 23 on 32 processors (right). The results of the new code version have a slightly better scaling property compared to the original code version and are more than 50 % faster due to the single processor optimization.

After porting the new code version to IBM BlueGene/L, the same simulations have been done on an IBM BlueGene/L system (see Fig. 3). Due to a much better ratio between compute power to communication capacity the results show a significantly better scaling property. A speedup of 15 on 16 processors (left) and 28 on 32 processors (right) has been achieved. Hence, it is feasible to run the $32 \times 128 \times 32$ grid case on 32 BlueGene/L PowerPC 440d processors which is not an optimal choice on an IBM p690 node with POWER4 1.3 GHz processors (compare Fig. 2 on the right). As the MPI communication of the parallelization in the s direction seems to be quite optimal a further speedup of the code seems to be unlikely. Instead it becomes obvious that an additional direction of parallelization has to be implemented to run the GEM code on larger number of processors.

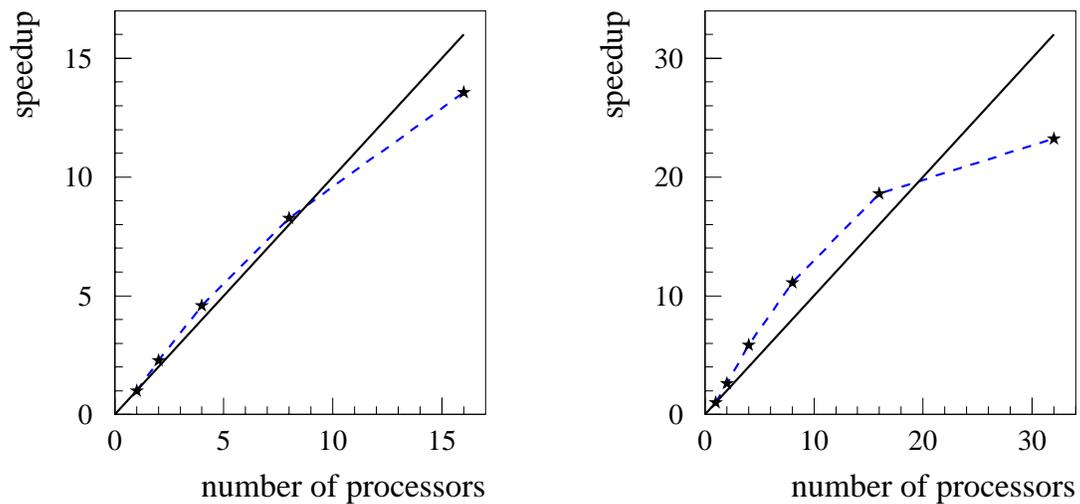


Fig. 2 Strong scaling of GEM code on an IBM p690 node with POWER4 1.3 GHz processors at RZG; speedup over the number of processors with the new code version (dashed); on the left for a $64 \times 256 \times 16$ grid and on the right for a $64 \times 256 \times 32$ grid; the solid line depicts linear scaling

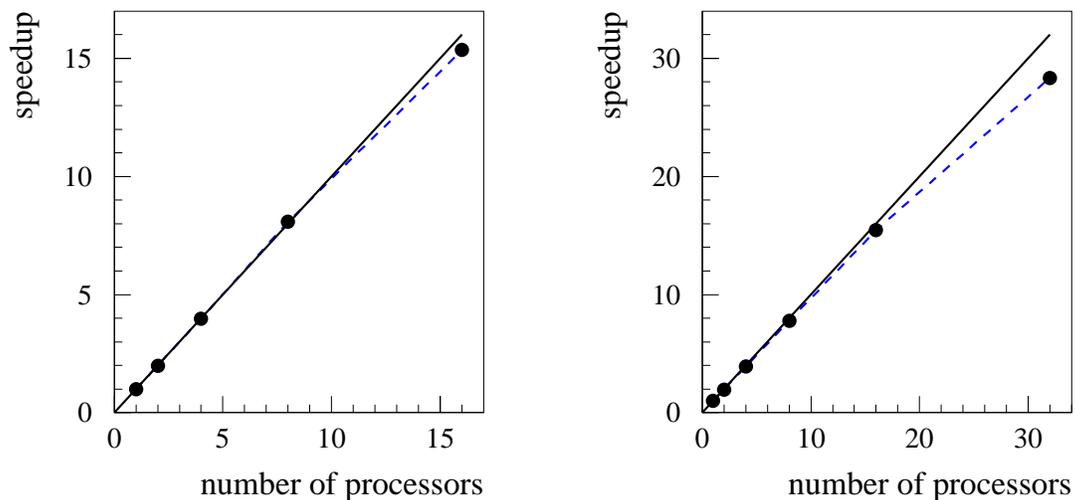


Fig. 3 Strong scaling of new GEM code version on an IBM BlueGene/L with PowerPC 440d 0.7 GHz processors at IBM Rochester centre; speedup over the number of processors with the new code version (dashed); on the left for a $64 \times 256 \times 16$ grid and on the right for a $64 \times 256 \times 32$ grid; the solid line depicts linear scaling

4.4 Concept for enhancement of scalability

The best choice for enhancement seems to be to parallelize in the x direction as the coordinate transformation in y direction is quite complicated to be parallelized. However, the data structure of the original code is not optimal for an x direction parallelization. All important arrays have the index structure (j,k,i) where i is the index of the x direction, j the index of the y direction and k the index of the s direction. The strong anisotropy of the physical problem along the magnetic field lines make more numerical operations necessary in the perpendicular plane (x and y coordinates) than along the magnetic field lines (s coordinate). As a matter of fact, parallelization in the x direction would be more intensively used than the parallelization in the s direction. Hence, a more optimal index structure would be (j,i,k) .

Another positive side effect of this index structure would be a better cache reuse. It can be clearly seen from Fig. 2 (right) that there is a super linear scaling for processor numbers of two, four and eight processors. This is caused by a poor single processor performance in the case of the $32 \times 128 \times 32$ grid. The reason is a cache size effect of the L2 cache of the IBM POWER4 processor which has 0.72 MB per processor and thus is unable to store the whole $32 \times 128 \times 32$ array of 1.05 MB simultaneously. Already for a two processor run the arrays fit again into the L2 cache memory as the arrays are half in size due to the memory distribution over two processors. However, this cache effect can not be seen for the BlueGene/L results shown in Fig. 3 (right) as the IBM PowerPC 440d has a sufficiently large L3 cache of 4 MB.

If the index structure is (j,k,i) and there are array operations such as $(j,k,i-1)$ and $(j,k,i+1)$ they could try to access memory which is not mapped to the L2 cache and has to be fetched on the IBM POWER4 processor from the next level L3 cache. Unfortunately, the most intensive numerical stencils of the differential operators act on the x and y directions and cause memory access with strides in the memory assigned to the i and j indices. In contrast there are no strides in the k index memory structure. Hence, an index order of the form (j,i,k) would restrict the strides on a smaller space of memory which would fit completely into the L2 cache memory for the grid sizes considered here.

The change of the index structure affects all parts of the code and is a laborious, time-consuming task which has to be done with great care. It is currently under way in close collaboration with the developing scientist, Bruce D. Scott, and will take probably a couple of weeks before the x direction parallelization will be addressed as the next step.

The x direction parallelization seems to be quite straight forward with two key issues: the additional distribution of guard cells and the sparse matrix solver parallelization both in x direction. The latter will be done in two stages. First, the parallelization of the tridiagonal solver of the GEM code; second, the parallelization of the more complicated time dependent sparse matrix structure of the next-generation code (GEMX) which is planned to be implemented by the Watson Sparse Matrix Package (WSMP) and/or the PETSc library. So finally, both more compute power and an enhanced physical model will be made available.

5 Proof of very high scalability of ORB5

ORB5 was tested by Cray on the Cray XT3 system (Jaguar) at ORNL for a strong scaling case up to 8192 (single core) processors (Fig. 4). The test case consisted of an Ion Temperature Gradient (ITG) driven simulation with a grid of $256 \times 256 \times 256$ cubic B-splines and a particle number of 512 million. Good scalability was demonstrated with a parallel efficiency of 70 % on 8k processors relative to 1k processors, for a fixed problem size requiring about 400 GB of main memory.

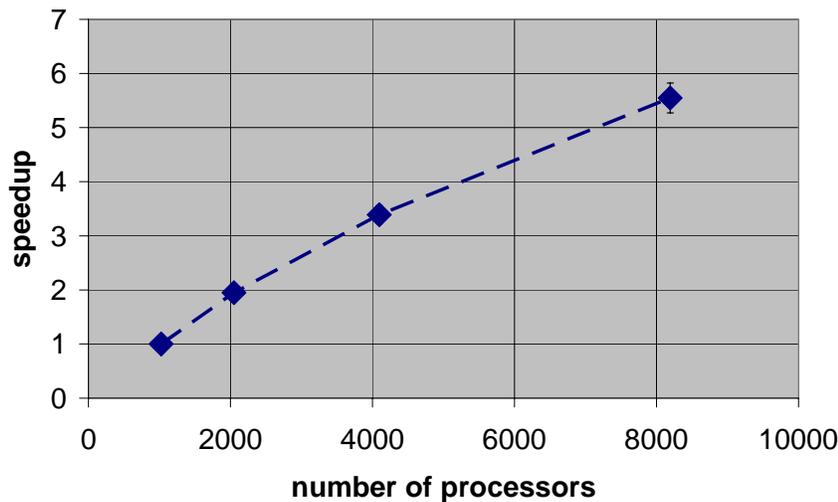


Fig. 4 Strong scaling of ORB5 for an ITG simulation with 5×10^8 particles with measurement points for 1k, 2k, 4k and 8k processors. All results were normalized on the 1024 processor result. (Measurements on Cray XT3 (single core) at ORNL; courtesy of Cray Inc)

A benchmark up to 8192 processors with a different test case of an Electron Temperature Gradient (ETG) driven simulation with a grid of $256 \times 256 \times 256$ quadratic B-splines and a particle number of 800 million was done on the BlueGene/L system at IBM Watson Research Centre. The results for a strong scaling are presented in Fig. 5 and show a high parallel efficiency of 88 % on 8k processors relative to 1k processors.

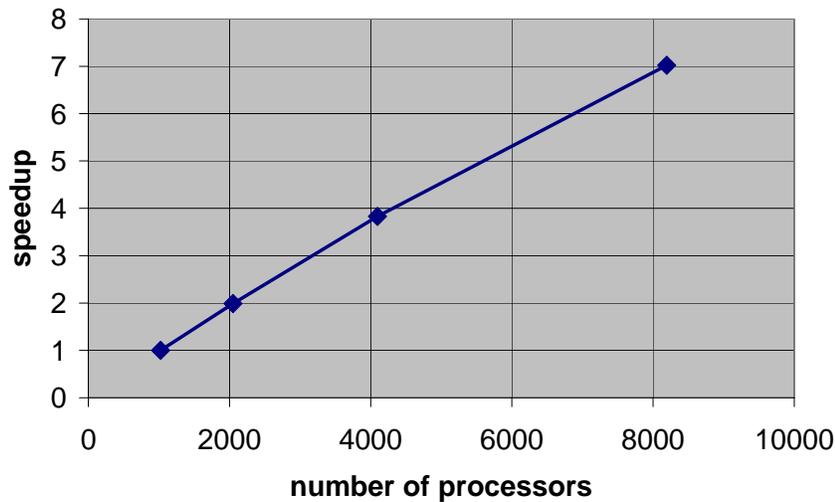


Fig. 5 Strong scaling of ORB5 for an ETG simulation with 8×10^8 particles with measurement points for 1k, 2k, 4k and 8k processors. All results were normalized to the 1024 processor result. (Measurements on BlueGene/L at IBM Watson Research Centre in co-processor mode due to memory limitation)

In addition, a weak scaling test for the same ETG simulation was performed, doubling the number of simulation particles with each doubling of processor numbers used, or in other words, fixing the number of particles per processor to approx. 800,000 (Fig. 6).

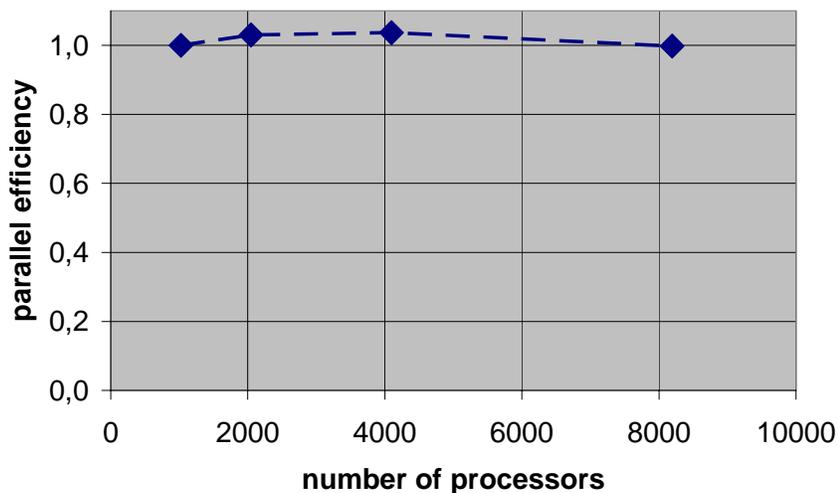


Fig. 6 Weak scaling of ORB5 for increasing problem sizes (with approx. 800,000 particles per processor) with measurement points for 1k, 2k, 4k and 8k processors. All results were normalized on the 1024 processor result. The parallel efficiency is slightly super-linear on 2k and 4k processors and one on 8192 processors. (Measurements on BlueGene/L at IBM Watson Research Centre in co-processor mode due to memory limitation)

In this way the memory could be fully exploited by always executing with the largest problem size possible per processor number. A slight super-linear behaviour was observed on 2048 and 4096 processors, the parallel efficiency approaching one again

on 8192 processors. This means that a 4 TB problem case can be efficiently treated without degradation on 8k nodes in co-processor mode, using one processor and all the memory per node for the calculations.

6 Proof of very high scalability of GENE

GENE was also tuned and tested for very high scalability, jointly with eSA4-T1 (eDEISA) (“Hyperscaling”). First GENE was also ported to BlueGene/L. First pre-runs with the previous version of GENE (GENE10) revealed that the good scalability started to degrade at 8 k processors.

For a follow-up version, a further dimension was parallelized. This new version was then tested during the so-called BGW days in late March 2007 at IBM Watson Research Center up to 32 k processors in weak scaling mode (fixed problem size per processor, i.e. increasing problem size with increasing number of processors), and up to 16 k processors in strong scaling mode (with fixed problem size for all processors). In both cases excellent scaling behaviour was observed.

From 1k to 16k processors, a parallel efficiency of 89% was achieved in strong scaling mode. In weak scaling mode, a parallel efficiency of 99% was achieved from 2k to 32k processors.

The enabling work description and the scaling results have been accepted for publication and will be presented at the CLADE 2007 workshop held in conjunction with the 16th International Symposium on High Performance Distributed Computing (HPDC-16) in Monterey Bay, US (June 25, 2007).

Publication:

H. Lederer, R. Hatzky, R. Tisma, A. Bottino, F. Jenko: **Hyperscaling of plasma turbulence simulations in DEISA**, accepted for publication by CLADE 2007

7 Further roadmap

Enabling of EUTERPE has been successfully finished in this reporting period. It has been recommended by a Swiss/German collaboration (CRPP/IPP) and will be needed for the DECI project Gyro3D from the DECI call of spring 2006.

The enabling work of the GEM code was requested by a multinational consortium from Austria, Germany and Portugal last summer. A single processor optimization with a performance increase by more than 50 % was achieved. This new code version has already been passed to the consortium and runs in production. The close collaboration will continue as scheduled in the project months 37 to 39 to implement the additional parallelization in the x-direction.

This winter, a new request was received from a multinational consortium from England, Germany and Switzerland with respect to the GYGLES [9] code. GYGLES (Gyrokinetic Global LinEar Solver) uses a linear time-evolution approach to simulate ITG (ion-temperature-gradient-driven) microinstabilities in toroidal geometry with a gyrokinetic model. It has very recently been extended to treat the evolution of global electromagnetic modes which had been an unresolved problem for PIC simulations over

the last decade. Consequently, the compute power needed for such simulations increased significantly due to the two scale problem of the fast electrons and relatively slow ions. Hence, it is planned to run the GYGLES code on several hundreds of processors instead of the typical number of 32 processors used now.

The institutions involved are:

the United Kingdom Atomic Energy Authority (UKAEA) Culham, UK
(Dr. Simon Pinches)

Max Planck Institute for Plasma Physics (IPP), Greifswald, Germany
(Prof. P. Helander)

Centre de Recherches en Physiques des Plasmas (CRPP), Lausanne, Switzerland
(Prof. L. Villard)

GYGLES enabling has been given priority over the earlier mentioned enabling of HAGIS code in agreement with its author, Simon Pinches. It has been scheduled in the project months 40 to 42.

The list of codes supported by JRA3 so far: TORB, ORB5, GENE, EUTERPE, GEM (and GYGLES in the future) are approaches to solve the first-principle gyrokinetic and gyrofluid equations.