

CONTRACT NUMBER 508830

DEISA
**DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR
 SUPERCOMPUTING APPLICATIONS**

European Community Sixth Framework Programme
 RESEARCH INFRASTRUCTURES
 Integrated Infrastructure Initiative

JRA3: Activity report and roadmap evaluation

Deliverable ID: DEISA-D-JRA3-5

Due date: October 31, 2006
Actual delivery date: November 22, 2006
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 Services) | |
| 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 TURBULENCE SIMULATION BY EUROPEAN PLASMA PHYSICISTS | 4 |
| 3 ENABLING OF EUTERPE STELLARATOR CODE | 4 |
| 3.1 Introduction | 4 |
| 3.2 Single processor performance optimization | 5 |
| 3.3 Optimization of the scaling properties | 7 |
| 3.3.1 Load imbalance | 7 |
| 3.3.2 The correct implementation of the domain decomposition | 8 |
| 3.3.3 The domain cloning concept and its implementation | 9 |
| 4 FURTHER WORK ON APPLICATIONS FROM PLASMA PHYSICS | 11 |
| 4.1 Supplement on ORB5 enabling | 11 |
| 4.1.1 Portability issues | 11 |
| 4.1.2 Performance issues | 11 |
| 4.2 Update of TORB code | 13 |
| 5 ROADMAP: FURTHER RELEVANT APPLICATION CANDIDATES FROM PLASMA PHYSICS | 14 |

1 Introduction

1.1 Executive Summary

This document “JRA3: Activity report and road map evaluation” is the 30 month deliverable DEISA-D-JRA3-5 for Joint Research Activities in Plasma Physics. A detailed description of the enabling work on the EUTERPE code [1] is given, with an emphasis on hyperscaling.

EUTERPE code was analysed in detail, and bottlenecks were identified. Performance enhancements were done at several levels. Originally, the usability of the code was limited to a maximum value of typically 64 processors. Through the enabling work, this restriction has been removed by application of a new parallelization strategy (domain cloning concept). For weak scaling, a parallel efficiency of 98% could be achieved by increasing the processor number from 64 to 512 processors.

In addition, a supplement on the ORB5 code [2] has been provided improving its scalability property. A first software revision amending some minor bugs of the TORB code [3] has taken place. For dissemination purposes in the area of hyperscaling, a lecture on domain cloning has been given at the 1st DEISA Training Session [4].

1.2 References and Applicable Documents

[1] Jost, G., Tran, T.M., Cooper, W.A., Villard, L., and Appert, K.: Global linear gyrokinetic simulations in quasi-symmetric configurations. *Phys. Plasmas*, **8**: 3321, 2001

[2] Bottino, A., Angelino, P., Hatzky, R., Jolliet, S., Peeters, A.G., Poli, E., Sauter, O., Tran, T.M., and Villard, L.: Global Nonlinear Gyrokinetic Simulations in Tokamaks. *Proceedings of the EPS 33th conference on plasma physics*, Rome, 2006

[3] Hatzky, R., Tran, T.M., Könies, A., Kleiber, R., and Allfrey, S.J.: Energy Conservation in a Nonlinear Gyrokinetic Particle-in-cell Code for Ion-Temperature-Gradient-driven (ITG) Modes in theta-Pinch Geometry. *Phys. of Plasmas*, **9**: 898, 2002

[4] Hatzky, R., and Bottino, A.: Domain Cloning – A new approach to parallelization of particle-in-cell (PIC) codes, 1st DEISA Training Session, Paris, 2006

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

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

[7] Decyk, V.K., Karmesin S.R., de Boer A. and Liewer, P.C., Optimization of Particles-in-Cell Codes on RISC Processors, *Computers in Physics*, **10**, 290 (1996)

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

1.3 List of Acronyms and Abbreviations

| | |
|-----------------------|---|
| CEA | Comité de l'énergie atomique, see http://www-cad.cea.fr |
| 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 |
| Fortran | FORmula TRANslation, scientific programming language |
| IPP | Max-Planck-Institut für Plasmaphysik, Garching, see http://www.ipp.mpg.de |
| ITER | International Thermo-Nuclear Experimental Reactor |
| JRA | Joint Research Activity |
| 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/ |
| RZG | Rechenzentrum Garching, see http://www.rzg.mpg.de |
| SMP | Symmetric MultiProcessor |
| UKAEA Culham | Fusion power, see http://www.fusion.org.uk/ |
| Universität Innsbruck | Plasma and Energy Physics, see http://www.uibk.ac.at/th-physik/pep/ |
| Wendelstein 7-X | Fusion experiment, see http://www.ipp.mpg.de/ippcms/eng/pr/publikationen/W7X_engl.pdf |
| WSMP | Watson Sparse Matrix Package, see http://www-users.cs.umn.edu/~aqupta/wsmp.html |

2 Turbulence simulation by European Plasma Physicists

According to plasma physicists, a challenging scientific goal is the “numerical tokamak and stellarator”, in which the simulation of microscopic turbulence covers in a single run the whole plasma region and a time interval corresponding to global profile adjustment (“energy confinement time”), i.e. a few seconds in a reactor, with sub-microsecond time steps. At the same time the predictions of these and other models should satisfy accuracy requirements making them useful for engineering applications (like the design of a power plant).

Simulation of plasmas in tokamak geometry is typically two-dimensional in configurational space due to the symmetry in toroidal direction. Hence, simulation of plasmas in full three-dimensional stellarator geometry is in general more demanding as the simulation domain has to take into account this fact. Usually, the discretization of the spatial three-dimensional model equations, as for example in the EUTERPE [1] code, leads to systems of coupled equations that naturally become very large. Nevertheless, such simulations should be as realistic as possible, i.e. cover as much of the physics as possible. Thus, increased supercomputing power gives the possibility to implement more physics.

The computational power assigned to the EUTERPE code usage by DECI will be used to run for the first time gyrokinetic nonlinear plasma turbulence simulations globally in full three-dimensional geometry of a plasma device such as the stellarator Wendelstein 7-X.

3 Enabling of EUTERPE Stellarator Code

3.1 Introduction

The EUTERPE [1] code solves the linear electrostatic gyrokinetic equation for ions (the electrons are assumed to be adiabatic) in the whole plasma domain (full radius) for a three-dimensional realistic stellarator geometry. It is presently the only code with such capabilities. Other codes are either restricted to axisymmetric configurations or use as their simulation domain a flux tube which covers only a small portion of the plasma. EUTERPE was developed at CRPP-EPFL (Lausanne) and subsequently has been extended and used at the IPP (Greifswald). The ongoing development is done in collaboration between both institutes.

Numerically the code uses a Monte Carlo method, the so-called particle-in-cell (PIC) method, to follow the characteristics of the gyrokinetic partial differential equation. In order to decrease the statistical noise a δf -approach, which is equivalent to a control variates method, is used. This method serves to reduce the noise by orders of magnitude. The charge assignment process and the discretization of the three-dimensional Helmholtz equation have been unified by employing tensor product B-splines as finite elements. The resulting very large system of equations is solved using the PETSc library which simplifies parallelization of the sparse matrix operations and provides various parallel iterative solvers and preconditioners. The parallelization strategy consists of a domain decomposition concept in the toroidal direction. It has been implemented with the Message-Passing Interface (MPI library) which is also supported by the STS library. The use of the STS library for stellarators makes a high resolution - especially in the angular directions - and a correspondingly large particle number mandatory.

Consequently simulations for stellarators are computationally much more expensive than for tokamaks. For example, for a linear calculation of the stellarator Wendelstein 7-X, a resolution of at least 64X64 grid points in the angular directions and 64 grid points in radial direction is necessary. Due to different modes with similar growth rates and their coupling through the complex equilibrium structure very often stellarator configurations require a huge number of time steps until a robust linear mode structure has consolidated.

The EUTERPE code has been typically run on 32 to 64 processors. In this range it shows good scalability properties. In general, good scalability of PIC codes can be achieved even for thousands of processors e.g. with the TORB [2] code which solves the nonlinear gyrokinetic equation in a θ -pinch configuration. Due to a recently developed parallelization concept called domain cloning (see Refs. [5] and [6]) the TORB code scales nearly optimal and shows a parallel efficiency of more than 90 % on 4192 processors. The goal is to benefit from this new parallelization concept and to improve significantly the scalability of the EUTERPE code.

3.2 Single processor performance optimization

The enabling work on the EUTERPE 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.

The EUTERPE code needs to assemble the matrix which follows from the B-spline discretization of the potential equation. The matrix has to be assembled in a pilot run and is stored in files afterwards. As long as the equilibrium and phase factor do not change it can be used for several simulations. The assembly typically takes less than 15 Min and therefore, its performance is only of minor importance. Instead, we will concentrate on the code part being executed for the simulation.

At the beginning the code is initialized (`init`) which takes less than 5 Min. Afterwards the code loops over just one time step (`mainloop`). Usually, a full simulation consists of thousands of time steps. Hence, the time spent in the initialization is negligible and no performance improvements have been done on this part of the code.

As the EUTERPE code belongs to the family of PIC codes, there are some strategies [see Ref. 7] to speed-up such codes. Some of these had been already applied to increase the single processor performance of the ORB5 code (see DEISA-D-JRA3-4). However, the single processor performance of the original EUTERPE code version was only 235 Mflop/s for the main loop (see Table 1) which shows that much more had to be done on its structure than on the ORB5 code.

| Subroutine | #calls | Inclusive | | | Exclusive | | |
|------------|--------|-----------|-------|---------|-----------|------|---------|
| | | Time(s) | % | MFlops | Time(s) | % | MFlops |
| init | 1 | 241.919 | 100.5 | 137.095 | 30.686 | 12.7 | 44.318 |
| initfiel | 1 | 9.590 | 4.0 | 0.442 | 9.590 | 4.0 | 0.442 |
| initions | 1 | 166.673 | 69.2 | 134.752 | 166.673 | 69.2 | 134.752 |
| mainloop | 1 | 240.795 | 100.0 | 234.970 | 0.119 | 0.0 | 115.087 |
| onestep | 1 | 240.676 | 100.0 | 235.030 | 0.001 | 0.0 | 0.006 |
| setrho | 3 | 92.536 | 38.4 | 285.751 | 5.577 | 2.3 | 0.014 |
| poisson | 3 | 7.924 | 3.3 | 153.049 | 0.163 | 0.1 | 44.135 |
| push | 2 | 175.185 | 72.8 | 218.359 | 162.358 | 67.4 | 233.920 |
| parmove | 2 | 12.827 | 5.3 | 21.384 | 12.827 | 5.3 | 21.384 |

Table 1 Performance output (processor 16) of the original EUTERPE code for a parallel 64 processor run of just one iteration

In the following, we list the most effective performance changes which result in a performance increase of a factor of two in the main loop Mflop rate (see Table 2):

- a) We have eliminated array copying by the compiler into temporary arrays before passing them to a specific subroutine or function. This copying is sometimes unnecessary but done anyway for safety reasons (to achieve better compiling reliability) when arrays with Fortran 90 syntax are passed to C routines such as the MPI library routines.
- b) We have introduced guard cells in the periodical chi direction of the potential and its derived quantities to prevent costly MODULO calculations.
- c) We have swapped in several arrays the order of indices to get an optimal sweep for cache reuse. In addition, copying of some arrays becomes obsolete as they have already perfect structure for communication buffers.
- d) The different attributes of the particles had been stored in separate arrays. However, in the code there are many loops over all particles. Hence, for better cache reuse it is more efficient to store the particle attributes in a two dimensional array with the first index over the attribute and the second index over the particle number. It is especially helpful when several attributes of one particular particle are needed for calculations inside the same particle loop.
- e) When particles leave a specific domain they are communicated to the processor belonging to the domain the particles reside on next. This had been done originally for each particle attribute separately. Now, the particle is communicated with all its attributes at the same time. Hence, both sorting of the particles to be moved (`parmove`) and their communication becomes more efficient.
- f) After the implementation of items d) and e) it is quite simple to implement a cache sort algorithm. It sorts the Monte Carlo particles relative to their position to the grid cells of the electrostatic potential. Hence, a very high cache reuse of the electrostatic field data is achieved which significantly improves the Mflop rates of the routines `setrho` and `push`. The overhead caused by the sort routine itself (`isort`) had been minimized. A switch has been implemented to give the choice to enlarge a work array for the sorting process which can speed up the `isort` routine by a factor of three. Usually the resident memory size of the simulation is small enough to take advantage of this new feature.
- g) Some loops had been programmed with variable loop bounds. Hence an optimization of the loop structure was not possible for the compiler at compile time. We have introduced CASE SELECT constructs with fixed array bounds
- h) ~~Instead~~ nested loops have been unrolled by hand.

| Subroutine | #calls | Time(s) | Inclusive | | Time(s) | Exclusive | |
|------------|--------|---------|-----------|---------|---------|-----------|---------|
| | | | % | MFlops | | % | MFlops |
| init | 1 | 185.116 | 138.1 | 140.841 | 11.101 | 8.3 | 136.071 |
| initfiel | 1 | 19.750 | 14.7 | 0.216 | 19.750 | 14.7 | 0.216 |
| initions | 1 | 112.879 | 84.2 | 129.528 | 112.879 | 84.2 | 129.528 |
| mainloop | 1 | 134.057 | 100.0 | 417.810 | 0.639 | 0.5 | 259.910 |
| onestep | 1 | 133.413 | 99.5 | 418.575 | 3.733 | 2.8 | 39.153 |
| setrho | 3 | 72.031 | 53.7 | 370.448 | 3.193 | 2.4 | 0.088 |
| poisson | 3 | 6.384 | 4.8 | 147.145 | 0.075 | 0.1 | 16.141 |
| push | 2 | 67.692 | 50.5 | 542.706 | 67.692 | 50.5 | 542.706 |
| parmove | 2 | 6.695 | 5.0 | 6.359 | 6.695 | 5.0 | 6.359 |
| isort | 1 | 3.958 | 3.0 | 152.894 | 3.958 | 3.0 | 152.894 |

Table 2 Performance output (processor 16) of the new EUTERPE code for a parallel 64 processor run with particle cache sort

A major contribution of the performance speed-up (40 %) comes from the implementation of the cache sort algorithm for the Monte Carlo particles. This can be seen by comparing the main loop Mflop rates with (418 Mflop/s) and without (305 Mflop/s) the cache sort switched on (see Table 2 and Table 3).

| Subroutine | #calls | Time(s) | Inclusive | | Time(s) | Exclusive | |
|------------|--------|---------|-----------|---------|---------|-----------|---------|
| | | | % | MFlops | | % | MFlops |
| init | 1 | 177.374 | 97.5 | 143.617 | 11.109 | 6.1 | 135.964 |
| initfiel | 1 | 20.630 | 11.3 | 0.206 | 20.630 | 11.3 | 0.206 |
| initions | 1 | 112.550 | 61.8 | 129.918 | 112.550 | 61.8 | 129.918 |
| mainloop | 1 | 181.993 | 100.0 | 304.802 | 1.691 | 0.9 | 98.189 |
| onestep | 1 | 180.102 | 99.0 | 307.075 | 4.567 | 2.5 | 32.015 |
| setrho | 3 | 91.750 | 50.4 | 291.676 | 4.446 | 2.4 | 0.063 |
| poisson | 3 | 6.053 | 3.3 | 154.911 | 0.072 | 0.0 | 16.901 |
| push | 2 | 102.130 | 56.1 | 359.864 | 102.130 | 56.1 | 359.864 |
| parmove | 2 | 8.588 | 4.7 | 4.961 | 8.588 | 4.7 | 4.961 |

Table 3 Performance output (processor 16) of the new EUTERPE code for a parallel 64 processor run without particle cache sort

Finally, for better readability of the code we have collected all the calls to PETSc routines and most of the calls to MPI communication routines in specific modules.

3.3 Optimization of the scaling properties

3.3.1 Load imbalance

For a 64 processor run a slight load imbalance has been detected for a conventional simulation as can be seen by the dotted line of Fig. 1. In the gyrokinetic model the charge of the particles is projected in shape of a gyro-ring on the grid (scatter operation). Accordingly, the field acting on each particle is gyro-averaged over the gyro-ring (gather operation). However, depending on the position of the particle in the physical domain, the size of the gyro-ring can differ significantly relative to the grid resolution. The gyro-

ring could cover in some regions of the domain e.g. 10 grid cells but in other regions e.g. 50 grid cells. Thus, the scatter and gather operations would involve the memory locations of grid cells which would be for 50 grid cells on average much more distant than for just 10 grid cells. Hence, better cache reuse would become likely for equidistant points on relatively small rings but not for equidistant points on large rings.

Hence, the improved Mflop rate due to better cache reuse can be seen for processors where the average size of the rings is relatively small compared to the grid resolution. As a test, we have reduced the gyro-rings to their center points (see Fig. 1 dashed line) and the load imbalance disappears immediately. As long as the gyro-ring induced load imbalance is quite small (approx. $\pm 7\%$), it can be ignored as a minor effect whose origin is well understood.

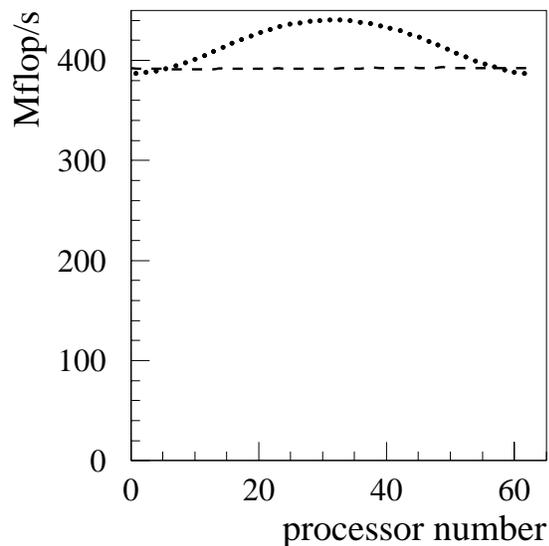


Fig. 1 The Mflop rate of the main loop for every processor of the full domain. Simulation with (dotted line) and without (dashed line) gyro-rings

3.3.2 Optimized implementation of the domain decomposition

For the domain decomposition concept different portions of the physical domain and of the corresponding electrostatic field grid are assigned to different processors together with the Monte Carlo particles that reside on them. The physical domain is split into several subdomains. As particles move from one region to another, they are communicated to the processor which is associated with the new region. A natural limitation is that at least one grid cell of the discretized field equation has to be hosted by each subdomain. Thus, the maximum number of subdomains is limited by the number of grid points (B-splines) in the direction of parallelization (toroidal direction). In addition, a certain number of guard cells depending on the order of the used numerical stencil have to be stored. In the original implementation of the domain decomposition concept in EUTERPE had an unnecessary limitation. Depending on higher order of the B-splines discretization more than one grid point per subdomain were mandatory, e.g. for quadratic order two and for cubic order three grid points.

The reason becomes only transparent in context of the flowchart of the codes used before a EUTERPE simulation had been started:

Equilibrium code VMEC → Equilibrium coordinate transformation code → Equilibrium data file code → EUTERPE code

The data file code wrote the equilibrium data files for all the processors. Each data file contained the equilibrium data needed for the subdomain hosted by the according processor including just one guard cell. Hence, this information was only sufficient for a linear B-spline discretization. Already for quadratic B-splines there was a lack of information which could have been only overcome by the information residing in the neighboring equilibrium files for the other subdomains.

The flow chart has been simplified in the following way:

Equilibrium code VMEC → Equilibrium coordinate transformation code → EUTERPE code

Now the coordinate transformation code already writes out the equilibrium files in the following structure: the three-dimensional equilibrium is split in the toroidal direction, which is also the direction of parallelization, in as many files as there are grid points. When the equilibrium information has to be read in by the EUTERPE code each processor reads in the files providing the equilibrium information needed for each subdomain. Equilibrium information needed for the B-spline guard cells of the subdomains is communicated via MPI from neighboring processors. Thus, there is no more need for an equilibrium data program and each subdomain can host only one grid point in the direction of parallelization. As a consequence, the number of processors working in parallel on a toroidal 128 grid point equilibrium could be increased by a factor of two for quadratic and a factor of three for cubic B-spline discretization.

3.3.3 The domain cloning concept and its implementation

In analogy to the TORB and ORB5 code the domain cloning concept has been implemented in the EUTERPE code for an optimization of the scaling property and a decoupling of the selectable grid resolution from the number of processors used for the simulation. It is a relatively new parallelisation concept which gives the opportunity to run PIC codes in the range of thousand processors which is of special interest for running them on DEISA hardware [see Ref. 6].

The domain cloning concept is a combination of particle decomposition and domain decomposition including both of them as limiting cases. According to this, which was introduced by Ref. [5], the simulation domain is cloned, i.e. multiple copies of the same domain are made. The processors are divided into a number of groups equal to the number of clones. Each group of processors is assigned to one of the domain clones. And each domain clone is loaded with its own set of particles and a one-dimensional decomposition is performed such that each processor has a corresponding subdomain.

The domain cloning concept has been implemented in the EUTERPE code so that the number of processors can be independently chosen from the number of equilibrium points. For example, the equilibrium of 64x128x128 chosen for our benchmark has 64 points in radial, 128 points in poloidal and 128 points in toroidal direction. The toroidal direction is the direction of parallelization in the domain decomposition concept. For this concept only a maximum number of 128 processors can be used for parallelization. However, introducing two clones already doubles the total number to 256 processors. For a simulation run over one time step and an equilibrium of 64x128x128 grid points a main loop wall clock time of 134.1 s with 64 processors has been achieved (see Sec. 3.2). In Fig. 2 we show the soft scaling property normalized to the 64 processor main loop wall clock time evaluated on the IBM POWER4 1.3 GHz super computer of RZG.

For each doubling of the processor number the number of Monte Carlo particles had been accordingly doubled. The scaling shows nearly optimal properties (98 %) up to 512 processors which had been run with four domain clones and a total number of $2^{30} = 1,073,741,824$ particles.

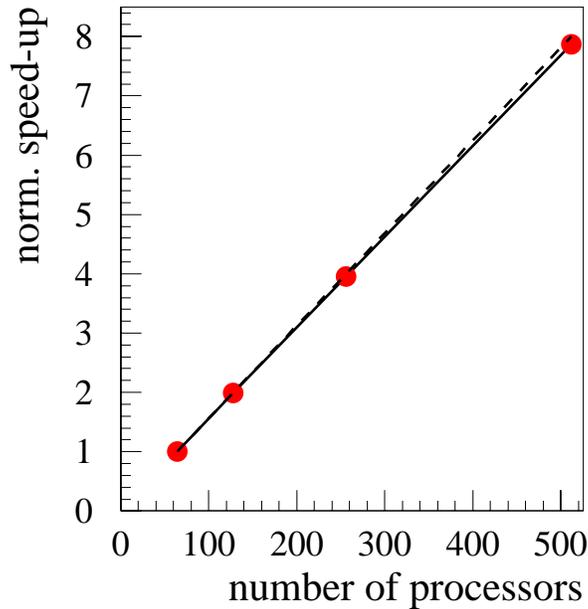


Fig. 2 Weak scaling of the EUTERPE code on the IBM POWER4 1.3 GHz super computer of RZG; squares: measured scaling, normalized to the 64 processor run; dashed line: linear scaling

Although a number of $\approx 2^{30}$ particles is necessary for nonlinear PIC simulations in a stellarator configuration, the scenario of linear PIC simulation of the original EUTERPE code needs much less particles. In such simulations the solver part (`poisson`) is more pronounced compared to the particle routines `setrho` and `push` of the code. The solver in the current version of EUTERPE has been so far parallelized over each clone with the consequence that each clone does still redundant work by solving the same field equation. As the near-perfect scaling of Fig. 3 shows, this is sufficient for the very large particle numbers needed for turbulence simulation but not for the much smaller particle numbers needed for linear simulation. As a result of this sequential part on the level of the domain clones, a scaling for a fixed number of markers (strong scaling) will give not optimal results for small numbers of particles.

Hence, the developing scientists of the EUTERPE code would like to have a better strong scaling property for the original linear simulations. They have requested to parallelize the solver over all processors by defining a new communicator for the PETSc library and by distributing the matrix over all processors. Thus, a supplement to the enabling work for the EUTERPE code is scheduled in the next deliverable with a focus on the evaluation and improvement of the strong scaling property

4 Further work on applications from plasma physics

4.1 Supplement on ORB5 enabling

The scientists developing ORB5 made a request for additional enabling work. In the framework of JRA3 activities we have complied with the request providing a supplement to the original enabling work done for deliverable DEISA-D-JRA3-4.

4.1.1 Portability issues

The portability of the ORB5 code has been further enlarged. The code now also runs on the Intel Itanium2 processors used in the SGI-Altix-System of LRZ. In addition, the code has been ported to the MareNostrum super computer (IBM Power PC 970FX processors) at BSC. Tests with 256 and 512 processors have been successfully achieved. The IBM WSMP library had to be provided especially for this architecture which had been done in close collaboration with its IBM developer Anshul Gupta.

Thus, ORB5 code has been given the portability that it could be used for the corresponding DECI 2005 project flexibly in the whole DEISA environment.

4.1.2 Performance issues

Memory

As it is planned to run ITG simulations of very large configurations, the memory consumption of the solver is a key issue. A new feature has been implemented which gives the possibility to construct the matrices and store them on disk. The benefit is twofold. On the one hand, matrices have to be constructed only once and can be reused for several simulations with the same equilibrium conditions. On the other hand, there is a clear cut between the construction of the matrices being part of the initialisation phase and the simulation phase of the ORB5 code. Hence, memory fragmentation caused by memory allocation and deallocation while building the matrices does not affect the simulation phase where a maximum of contiguous memory is needed. In addition, arrays used only temporarily have been shifted from the heap to the stack further minimizing the risk of the build-up of memory fragmentation.

The IBM WSMP routines were also optimized for minor memory consumption by adapting them to the special needs of the ORB5 code. Usually a permuted copy of the matrix is saved for iterative refinement of the solution. In the new version of the IBM WSMP library user's control has been enlarged to suppress the construction and saving of such a copy of the matrix if no iterative refinement is planned. For the ORB5 code refinement is not necessary. Thus, the memory consumption of the WSMP routines has been significantly reduced.

Scalability

As a result of DEISA enabling, the ORB5 code had used ScaLAPACK routines for solving the dense matrix and WSMP routines for solving the sparse matrix. Although ScaLAPACK is faster when solving dense matrices on very few processors, e.g. one or two, its scaling property is not as good as it is for IBM WSMP. This is of special interest as this part of the program is scaling for both ScaLAPACK and IBM WSMP only to a few

tens of processors. For runs with many thousands of processors it behaves like a “quasi sequential” part and any improvement on its performance would have direct impact on Amdahl's law.

Fig. 3 shows the speed-up of IBM WSMP execution time relative to ScaLAPACK execution time as a function of the number of processors used. The three curves represented different matrix sizes for grids with 64x64x64 (magenta), 126x128x128 (blue) and 256x256x256 (red) grid points. The smaller the number of grid points the smaller the maximum number of processors which can be used in parallel. Although the IBM WSMP solver was originally developed for sparse matrices its scaling property is superior to the block matrix solver from ScaLapack if the number of processors is sufficiently large. For the case of 256x256x256 grid points and 32 processors the IBM WSMP solver is faster by a factor of more than three.

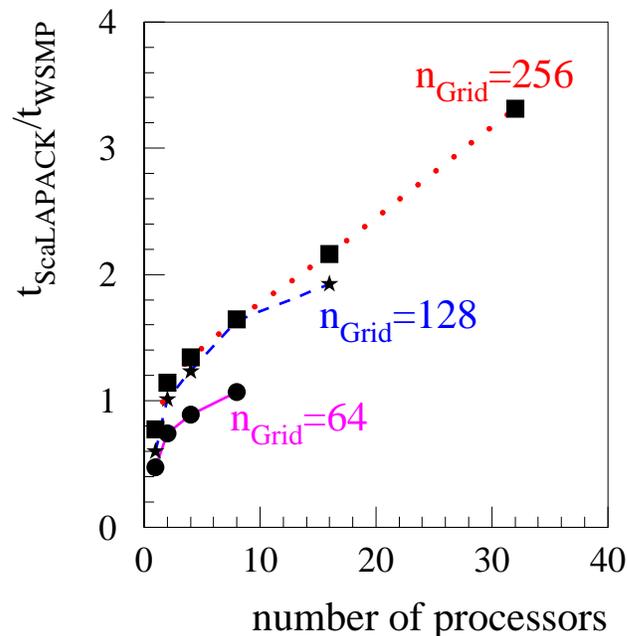


Fig. 3 Speed-up of IBM WSMP execution time relative to ScaLAPACK execution time as a function of the number of processors. The three curves represented different matrix sizes for grids with 64x64x64 (magenta), 126x128x128 (blue) and 256x256x256 (red) grid points

However, in the past a solving of both matrices with IBM WSMP was prevented by not having the opportunity of switching to different MPI communicators during run time. This obstacle has been removed within close collaboration with the IBM developer Anshul Gupta. A change of the MPI communicator during run time is now implemented so that both matrices are solved by the IBM WSMP library which improves the overall scalability of the code.

Increasing the maximum number of particles

Large simulation runs can consist of a number of Monte Carlo particles larger than $2^{31}=2,147,483,648$. The 2^{31} particle limit is critical for the programming structure of the ORB5 code as the standard 4-Byte Integer representation of FORTRAN can not handle larger integer numbers without causing an overflow. Hence, the parts of the code relying

on 8-Byte Integer arithmetic had to be adapted. A general change to 8-Byte integers would have been more laborious because of the changes involved for calls to the Message Passing Interface (MPI) library interfaces which were programmed originally in ORB5 for 4-Byte Integers.

4.2 Update of TORB code

A software revision of the turbulence code TORB had been necessary as the developing scientists have discovered some minor bugs. The code version used for DEISA had been released as (probably) stable version 1.22. It already differs significantly from the version 1.23 currently under development. As it is of importance to have a most stable version for DEISA, the decision was taken to keep the version 1.22 but correcting the bugs discovered so far. However, a simple copy and past of the source code of the amended parts was not possible. Instead, the version 1.22 had to be adapted by hand to catch up with a subset of the changes made in the new version 1.23. After modification the corrected code version 1.22 was tested and released to the Common Production Environment of DEISA.

5 Roadmap: Further relevant application candidates from plasma physics

A screening for important applications from plasma physics to be supported in DEISA has taken place. Among these institutions are CEA/Cadarache, UKAEA/Culham, CRPP/Lausanne, IPP/Garching and IPP/Greifswald.

Enabling of EUTERPE has been the major task in the past reporting period. EUTERPE 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. Supplementary support has already been asked for (see Sec. 3.3.3).

Elaboration of code candidates has been done in contact with the institutions named above.

This summer a request has been received by a multinational consortium from Austria, Germany and Portugal with respect to support for the GEM code. GEM simulates the gyrokinetic equations for electrons and ions with an electromagnetic gyrofluid model [8]. It is a global model in which the dependent variables are allowed to have arbitrarily varying zonal profiles (flux surface averages), although the parameter set is local, with only quadratic nonlinearities retained. The model is well suited for turbulence and turbulent transport studies in the boundary region of tokamak plasmas, where ITG drift Alfvén and interchange turbulence are believed to play a relevant role.

The institutions involved are:

Centro de Fusão Nuclear (CFN), Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal (Prof. Fernando Serra)

Institut für Theoretische Physik der Universität Innsbruck, Austria
(Dr. Alexander Kendl)

Max Planck Institute for Plasma Physics, Garching, Germany
(Dr. habil. B.D. Scott, P.I.)

The HAGIS code, mentioned earlier as also being a candidate for application enabling, has been postponed after GEM enabling.

A first inspection of the complex enabling work for GEM leads to the estimate that it will completely cover the project months 31 to 42, together with the supplementary work for EUTERPE.

All the codes supported by JRA3 so far: TORB, ORB5, GENE, EUTERPE (and GEM in the future) are approaches to solve the first-principle gyrokinetic equations. For this purpose, different numerical models and methods are used to simulate plasma turbulence within a large variety of configurations such as θ -pinch, tokamaks and stellarators. These codes provide new instruments of numerical modelling, and promise qualitative progress in the theoretical understanding of turbulence and ultimately also in our capability to quantitatively predict its consequences as e.g. for the experiment ITER.