*33rd EPS Conference on Plasma Phys. Rome, 19 - 23 June 2006 ECA Vol.**30I**, P-4.153 (2006)*

Performance enhancements to ELMFIRE gyrokinetic code leading to new ranges of application

F. Ogando[1,2], A. Signell[3], J.A. Heikkinen[4], M. Aspnäs[3],
S. Henriksson[1], S.J. Janhunen[1], T.P. Kiviniemi[1]

[1] *Euratom-Tekes Association. TKK, Espoo, Finland.*

[2] *Universidad Nacional de Educación a Distancia, Madrid, Spain.*

[3] *Åbo Akademi, Turku, Finland.*

[4] *Euratom-Tekes Association. VTT, Espoo, Finland*

Introduction

Plasma turbulence has shown to be responsible for the so called anomalous transport that compromises the magnetic confinement of fusion plasmas. This anomalous transport takes values hundreds of times higher than predicted by the neoclassical theory. The study of the dynamics of turbulent plasmas has become a key issue in order to control the generation of turbulence in a plasma and its influence on the degradation of confinement.

With that purpose the ELMFIRE [1] code was developed as a joint project between VTT and TKK. It is a Monte Carlo full-*f* nonlinear gyrokinetic plasma simulation code in five-dimensional toroidal geometry. The code includes several physical features like multiple different species, collisions between them and neutral particle ionization. The movement equations take into consideration the existing drifts in tokamak plasmas.

The gyrokinetic model simplifies the description of motion of charged particle motion by averaging it over the quick gyration around the magnetic fields lines. This produces a reduction of phase space to five dimensions and thus reducing the computational requirements for its solving. The full-*f* formulation of the problem includes solving the whole five dimensional particle distributions, opposite to the delta-*f* technique, which for further simplification considers perturbations from local Maxwellian distributions. However the full-*f* technique included in ELMFIRE produces accurate results even in the presence of strong gradients or perturbations in the plasma.

Computational challenges

ELMFIRE defines and follows the dynamics of a big number of particles representing ion or electron gyrocenters and follows their dynamics in a fixed magnetic background and consistent electric field. From an initial particle distribution following given profiles an electrostatic potential is calculated using Sosenko's [2] model. Particles are then moved in the electrostatic

field with implicit consideration of electron parallel acceleration. Boundary conditions are set acconrdingly to the case under simulation for both particle motion and electrostatic potential field. The operation of ELMFIRE presents clearly identified peaks of resource requirements that produce limitations on the capability of solving large scale physical systems.

Regarding computational calculus, the advance of particles in the electrostatic field takes the most of time. However this task has been parallelized in an almost perfect way by distributing the particles into different parallel processes, with little or no overheads.

The electrostatic potential is calculated from density samples calculated from the gyrocenter distribution in reduced (5D) phase space. Once the equation is discretized in a three dimensional grid, potential is solved as the solution of a linear system with a sparse matrix. The matrix coefficients are cumulatively calculated from gyrocenter parameters, resulting in hundreds of non-zero coefficients per grid cell. The resolution of the linear system is parallelized by splitting the matrix into different processes using standard numerical tools (PETSc, PESSL,...) but as this process is of different nature as the particle distribution, particles of one process may affect matrix coefficients stored in all other processes.

The most straightforward solution applied so far was constructing this matrix with room for all expected non-zero values (many more than the actual non-zeroes) independently in every process and making a summation after processing all particles. However this posed the limitation of needing a huge storage proportional to the whole problem size. This automatically breaks scalability to very big systems in massively parallelized environments.

Here we present the AVL system developed and implemented to overcome the limitation imposed by memory and domain decomposition to reduce the time of network use.

AVL system

AVL is a binary-tree system developed and implemented into ELMFIRE to retain unlimited scalability during the potential resolution. Considering it is the most memory demanding phase, this provides the code good scalability to high number of processors.

AVL collects the contributions from particles to coefficient matrix and, instead of constructing with them a local matrix, classifies them accordingly to their corresponding processor into different memory buffers of limited size. Once a buffer is filled up during the particle processing, it is sent to the proper destination process using non-blocking process communication, included in the MPI communication protocol. This non-blocking communication allows network communication simultaneous to the normal advance of the code calculations. While the memory buffer is being transferred a secondary is used for storing new coefficients being calculated.

A system like the previous would limit by itself the memory requirements to a fixed value

regardless of problem size, at the expense of increasing the network communications. This increase would be enormous, since the contribution of different particles to same matrix coefficients are sent as different by network, producing overheads of the order of tens of times more communication (the number of particles contributing to a certain coefficient). AVL system introduce two key ideas to strongly reduce the network communication, producing not only scalable algorithm, but in some cases even faster than the original one due to reduced network communication.

- Use of searchable buffers, so that contributions from different particles to the same coefficients can be added before being sent. The fastest searchable structure is the binary-tree, basis of the AVL system.

- Consecutive particles initially tend to be close to each other, so that keeping track of last particles can strongly speed up the search operations. Hash cache tables have been designed for that purpose.

The performance of the AVL system is sensitive to the memory buffer size. High reduction in memory consumption can be achieved but resulting in more frequent communication steps that reduce code performance. Number of steps and total data transferred can be estimated from the pessimistic assumption that particles contributions fill the non-zero part of matrix in a random order. In order to estimate the scalability to bigger problem, the calculation uses parametres that remain limited in every run: a) Coefficients affected per particle ($\mu \approx 16$), b) particles per cell ($\beta \approx 500$), c) number of nonzero coefficients per matrix row ($\gamma \approx 100$) which depends on the relation of Larmor radius to cell size, d) α fraction of local matrix (theoretical lowest limit of required memory) stored as send buffers, and being P the number of processors. With those parameters, the number of transfer steps is:

$$m \approx -\frac{\mu\beta}{\log(1-\alpha/P)\gamma P} \tag{1}$$

which for small buffersize becomes inversely proportional to it. It has to be remarked, that network latency and multiple transfers mostly degrade communication performance, and that is why m parameter works as performance indicator. The previous expression is obviously valid only with $m \geq 1$. For optimum performance and least memory use, the limit $m = 1$ is sought by chosing the proper α value.

$$\alpha_1 = P \cdot \left[1 - \exp\left(-\frac{\beta\mu}{\gamma P}\right)\right] \tag{2}$$

This expression is approximately P (where buffers size equals whole matrix size) saturating for $P = \beta\mu/\gamma \approx 80$ to that same value. Therefore optimal AVL performace ($m = 1$) takes as much

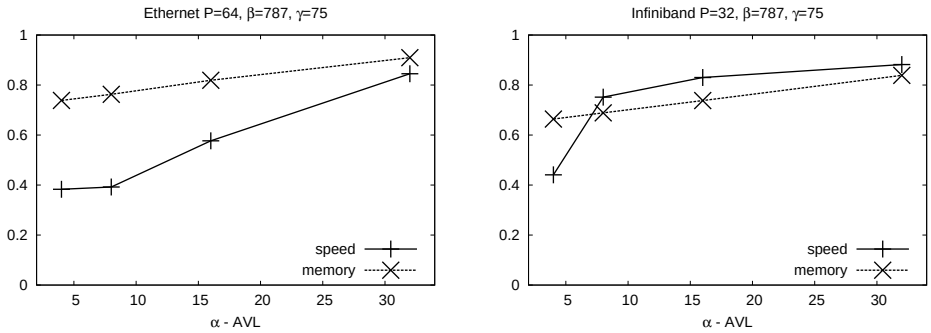
memory as traditional algorithm. However, for high number of processors, which in near future could be over 1000, presents a saturated optimum value that results in huge memory savings.

Domain decomposition

Domain decomposition [3] is a further step in parallelization of the system, in order to reduce the network communication which can be crucial factor in massively parallel runs. Domain decomposition divides the physical extent into several domains whose calculation is associated to a group of processes that keep track of all the particles inside it. It is still under implementation to ELMFIRE .

Results

The activation of the AVL has been a trade between computation time and network and memory use. That is why the optimal results are expected using slow network and big number of processors. The results shown here have been obtained using a cluster of HP ProLiant DL145 with gigabit ethernet connection at the CSC. The figure shows the dependency of memory consumption and computing speed for a medium sized case. Quantities have been normalized to the same run without using AVL.



AVL has shown to reduce the memory requirement of ELMFIRE to even less than 20% of original algorithm, although at the expense of losing speed. However AVL keeps scalability regardless of the processor number, being of extreme importance for the foreseen massively parallel “Millenium runs”.

Acknowledgements

The facilities of the Finnish IT Center for Science have been used in this work. This project has received funding from the European Commission.

References

[1] J.A. Heikkinen et al., J Comput Phys, **173**, 527 (2001)

[2] P.P. Sosenko et al., Physica Scr, **64**, 264 (2001)

[3] S. Ethier, SciDAC Workshop on Plasma Turbulence (2005).