Astronomical Data Analysis Software and Systems XII
ASP Conference Series, Vol. 295, 2003
H. E. Payne, R. I. Jedrzejewski, and R. N. Hook, eds.

FLY: a Tree Code for Adaptive Mesh Refinement

U. Becciani, V. Antonuccio-Delogu, A. Costa

INAF – Astrophysical Observatory of Catania

D. Ferro

Department of Physics and Astronomy – University of Catania

Abstract. FLY[1] is a public domain parallel treecode, which makes heavy use of the one-sided communication paradigm to handle the management of the tree structure. It implements the equations for cosmological evolution and can be run for different cosmological models. This paper shows an example of the integration of a tree N-body code with an adaptive mesh, following the PARAMESH scheme. This new implementation will allow the FLY output, and more generally any binary output, to be used with any hydrodynamics code that adopts the PARAMESH data structure, to study compressible flow problems.

1. Introduction

We have developed a powerful N-body code to evolve three-dimensional self-gravitating collisionless systems with a large number of particles ($N \geq 10^7$). FLY (**F**ast **L**evel-based N-bod**Y** code) is a fully parallel code based on a tree algorithm. It adopts periodic boundary conditions implemented by means of the Ewald summation technique. FLY is based on the one-sided communication paradigm to share data among the processors that access remote private data with any synchronism. The code was originally developed on a CRAY T3E system using the logically SHared MEMory access routines (*SHMEM*) and it was ported to SGI ORIGIN systems and IBM SP, on the latter making use of the Low-Level Application Programming Interface routines (*LAPI*). This paper shows an example of integration of a tree code with an adaptive mesh scheme. PARAMESH is a package of Fortran 90 subroutines using *SHMEM* and *MPI* libraries, designed to provide an application developer with an easy route to extend an existing serial code which uses a logically cartesian structured mesh into a parallel code with an adaptive mesh refinement. The computational domain is hierarchically subdivided into sub-blocks following a 3D tree data-structure. The use of *SHMEM* and the tree data-structure eases the integration with FLY, which adopts the same data structure and the same parallel communication library. This implementation of FLY with PARAMESH integrates the output of FLY

[1] `http://www.ct.astro.it/fly/`

Figure 1. Main FLY window and the window where the static parameter file is set.

with an adaptive mesh, having the same data structure as PARAMESH. The adaptive mesh structure can be read by FLY or generated from it, and contains the potential field of each data block of the mesh, following the PARAMESH scheme.

2. FLY Main Features

FLY is a dynamically load balanced code based on four main characteristics: it adopts a simple domain decomposition, a grouping strategy and a data buffering that minimize data communication. The domain decomposition is based on a fixed distribution of particles among the processors: the same number of particles is assigned to each processor. The data structures of both particles and tree are subdivided among the PEs to ensure a good initial distribution of the load and to avoid any bottleneck while accessing remote data. FLY uses a grouping strategy with the aim of computing a component of the force to be applied to all particles inside a *grouping cell* and to reduce the number of remote access to build the global force on each particle. With the data buffering, FLY uses the free RAM portion of each PE to allocate dynamically the tree and the bodies data structures in order to cache remote elements.

All these features allow FLY to run very large cosmological simulations on parallel systems with high performances: more than 20×10^4 particles per second were computed using 16 PEs on an IBM SP Power4 1300 MHz at Cineca.

3. From the Tree to the Adaptive Mesh

FLY is a pure gravitational tree N-body code performing LSS cosmological simulations using dark matter particles. Some codes include hydrodynamical effects, star formation as well as dark matter particles. However, due to the global code performances, the N-body number of particles that can be simulated is not very high. This obviously reduces the resolution of the formed structures.

The design of an interface between FLY and pure hydrodynamical codes allows us to have a high performance code running large simulations, with its data output (the potential field) integrated into the data structures of another code, running a gas evolution with the adaptive mesh refinement technique. The FLY output of the mesh is data input to the hydrodynamical code.

The mesh generated by FLY is identical to the PARAMESH structured mesh blocks. FLY creates the PARAMESH blocks at the minimum refinement level. Each block is geometrically mapped onto a tree cell. For each block all the sub-cells are computed: the *unk* variable being the potential field.

The computation is based on the tree structure, each block sub-cell is a virtual body and the unk variable is computed with the tree interaction (Barnes-Hut method). FLY can also decide that the block must be refined, depending on the mass density in each block, but it can also externally read all the blocks, at the refinement level given by the user. The FLY block of PARAMESH is the following:

```
TYPE BLOCK
 INTEGER(KIND=4)  :: lrefine_block
 INTEGER(KIND=4)  :: nodetype
 REAL(KIND=4), DIMENSION(ndim):: coord_block
 REAL(KIND=4) :: size
 REAL(KIND=4), DIMENSION(ndim) :: bnd_box_min
 REAL(KIND=4), DIMENSION(ndim) :: bnd_box_max
 REAL(KIND=4), DIMENSION(nxb+nguard,nyb+nguard,nzb+nguard):: unk
END TYPE BLOCK
```

FLY block can be used with any hydrodynamical code that adopts the PARAMESH data structure, to study compressible flow.

4. FLY User-friendly Interface

FLY has a graphical Tcl/Tk interface that help the user to create all the parameter files, excluding the initial condition file. The main window sets the working directory, the executable directory and will create directories if they don not exist. Figure 1 shows the main window and the window used to set the static parameter file. If this file exists, it is loaded with the values of the existing file. Other files are created using similar windows.

In the main window the user can click on the *Generate* button and insert data to create the fly_h.F module, the makefiles in the directory *src* and the script file that can be submitted to the system queue. The *Make* button executes the makefile and creates the executable program.

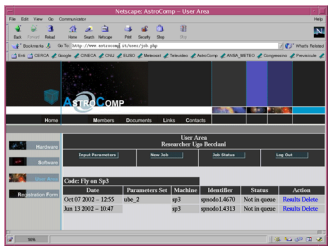

Figure 2. Monitor of Astrocomp job status

In the main window it is also possible to click on the buttons *Interactive Run* and *Batch Run* to submit the FLY job.

5. FLY in the Web

Astrocomp is a portal which creates a repository of easily usable computational codes and a common data base available to the entire international community. The Astrocomp server is based on a PHP-MYSQL environment. Registered users have free time on HPC systems included in Astrocomp and can run all the Astrophysical codes of the portal. FLY is included in this portal[2], where the users can run FLY directly on the IBM SP Power3 a 24 processor based system at the INAF OACT, on IBM SP Power4 a 512 processor based system and on the SGI Origin 3000 a 64 processor based system at the Cineca. The user can create parameter files with initial conditions and can submit a job on each system without knowing the commands of the selected system (Figure 2).

6. Conclusion

This new implementation of FLY is the first step towards the integration of this tree N-body code with hydrodynamical codes. The choice of an adaptive mesh refinement such as PARAMESH allows us to design a new interface aimed at *mixed* applications: a code running hydrodynamical simulations that receives data from FLY, evolving an N-body simulation at the same time. This kind of relationship could also be further enhanced using grid computing facilities.

References

Becciani, U. & Antonuccio, V. 2001, Comp. Phys. Comm. 54, 136

Barnes, J. & Hut, P. 1986, Nature 324, 446

MacNiece, P. et al. 2000, Comp. Phys. Comm. 126, 330

Di Matteo, P. et al 2003, this volume, 17

[2] `http://www.astrocomp.it`