Mem. S.A.It. Suppl. Vol. 9, 448

Memorie della
Supplementi

FLY-FLASH: A Software Interface for Adaptive Mesh Refinement - Treecode Simulations

M. Comparato[1], V. Antonuccio[1], U. Becciani[1], A. Dubey[2] T. Plewa[2] and D. Sheeler[2]

[1] Istituto Nazionale di Astrofisica – Osservatorio Astrofisico di Catania, Via S. Sofia 73, I-95125 Catania, Italy
[2] The ASC FLASH Center, The University of Chicago 5640 S. Ellis, Chicago, IL 60637
e-mail: mcomp@ct.astro.it

Abstract. We present an interface that allows us the execution of cosmological simulations by combining the capabilities of two different codes: FLY, a parallel treecode for N-Body simulations, and FLASH, a code for numerical hydrodinamic.This task is reached without heavy modifications of the codes, but by means of an interface that handles the communications between them. The underlying hypothesis is that the two codes are only loosely coupled, i.e. they interact only by exchanging the information to build the gravitational potential.

Key words. Cosmology: LSS simulations – Software: High Performance Computer

1. Introduction

In most cosmological simulations, the dominant source of the gravitational field is the Dark Matter (DM). Actually, the correctness of this statement depends on the spatial and temporal scales one is interested to: roughly speaking, gravity from the baryonic component starts to be a significant fraction of the global gravity on subgalactic ($r \leq 100\,kpc$) scales at $z \approx 0$ (within collapsed systems). At larger scales and for larger redshifts or within non-gravitationally-collapsing system, the contribution of the baryonic component to the global gravitational force is essentially negligible.
This means that, in most cases of relevance to cosmological simulations, one does not introduce significant errors if the gravitational force and potential are computed only for the DM component. This fact simplifies considerably the task of building a software interface between a parallel treecode and a eulerian AMR code. The baryonic (gaseous) component, modeled by the eulerian code, will then passively evolve under the action of the DM gravity. This means that the interaction between the two codes reduces to a periodic exchange of the gravitational potential, which should be sent from the N-body code to the eulerian one. The possibility of neglecting the gravitational back reaction of the baryonic component on the DM brings also another simplification: the two codes do not need to be run simultaneously, i.e. to be synchronized. Snapshots of the DM gravitational potential at some redshifts are the only input information required by the eulerian code to advance the

Send offprint requests to: M. Comparato

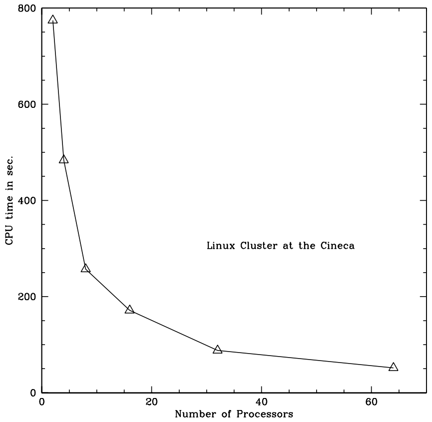

Fig. 1. CPU time in a run of FLY with 2 Million-particles.

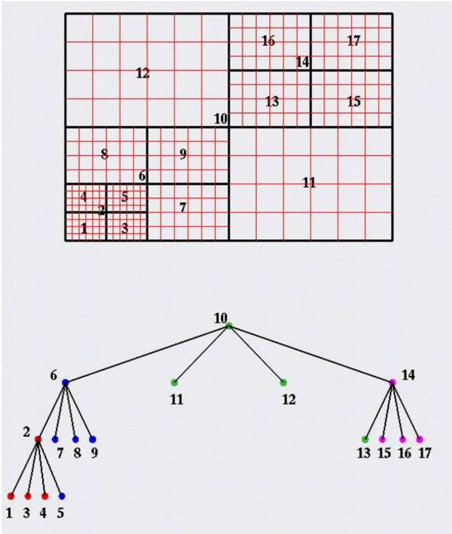

Fig. 2. A simple 2D example of a PARAMESH grid block tree covering a rectangular domain

simulation of the gaseous and stellar components.

2. FLY code

FLY (Antonuccio 2002) is a fully parallel code based on the tree Barnes-Hut algorithm for LSS cosmological simulations. It is based on the one-side communication paradigm to share data among the processors, thus enabling access to remote private data avoiding any kind of synchronism. The code runs on IBM SP by using the Low-Level Application Programming Interface routines (LAPI), and using the MPICH-2 library (Gropp 2005), FLY is also running on Linux Clusters. We developed the code using the Cineca IBM Linux Cluster system with 768 Intel Xeon Pentium IV Processors 3GHz, 512 KBcache, 2 PE per node, 788 GB Ram, and havinh the Internal Network: Myricom LAN Card ”C” Version and ”D” Version. The Fig. 1 show the FLY performance in a run with 64 Million-particles in this system

FLY adopts a very fast domain decomposition, dynamic load balance and data buffer to store locally remote data. A detailed description of the FLY code can be found at http://www.ct.astro.it/fly/

3. FLASH code

Flash (Fryxell 2000) is a modular, adaptive-mesh, parallel simulation code capable of handling general compressible flow problems found in many astrophysical environments. Until now Flash has mainly been used to address several problems related to thermonuclear flashes on the surfaces of compact stars. It is capable of handling the extreme resolution and physical requirements imposed by conditions in this explosions and to do so while making efficient use of parallel computer systems. The mesh of Flash is generated using PARAMESH (McNiece 2002). The structured grid blocks are distributed to cover the whole computational domain and the result is a tree of blocks as represented in the figure 2.

4. FLY-FLASH interface

The FLY-FLASH interface will allow the two codes to run independently and communicate by making use of the new process management features of MPI-2. This is the schema of the interaction in a FLY time-step: FLY makes itself available as a server, using two MPI routines. First it call MPI_OPEN_PORT to establish a port where it may be contacted, secondly it call MPI_PUBLISH_NAME that allows the server to publish a (port_name, service_name). The client application (the FLASH code in our case) will retrieve the port name from the service name with MPI_LOOKUP_NAME. This allows higher level of portability, with increasing levels of functionality. Then FLY waits for accepting connections from the client using the MPI_COMM_ACCEPT routine and receives the number of the interior cells (*nZones* parameter) of blocks of the PARAMESH structure that it will receive.

call MPI_OPEN_PORT(MPI_INFO_NULL, portName, ierror)

call MPI_PUBLISH_NAME('particleInterface', MPI_INFO_NULL, portName, ierror)

call MPI_COMM_ACCEPT(portName, MPI_INFO_NULL, 0, MPI_COMM_WORLD, flashComm, ierror)

call MPI_BCAST(nZones(1), ndim, MPI_INTEGER4, 0, flashComm, ierror)

After this preliminary communication between the server and the client, FLY evolves for a time-step. Due to the different physical phenomena involved in gravitational codes and fluid dynamics codes, one of the main issues is the coordination of the time-step evolution of the two codes. Before updating the system status, FLY sends the actual red-shift and the next red-shift of the nbodies evolution to the client. Then it waits for the total number of blocks that it will receive, and the bounding block coordinates.

call MPI_BCAST(totNBlocks, 1, MPI_INTEGER, 0, flashComm, ierror)

call MPI_RECV(bnd_box(1, 1), ndim * 2, MPI_REAL8, MPI_ANY_SOURCE, MPI_ANY_TAG, flashComm, istatus, ierror)

FLY, as server side of the communication, receives block coordinates and structure from FLASH, computes the mass distribution on the blocks and sends back to FLASH the results.

call MPI_SEND(blockCells(1, 1, 1), blockCellsSIZE, MPI_REAL8, istatus(MPI_SOURCE), 0, flashComm, ierror)

In the above MPI_SEND routine the blockCells array has the dimension of blockCellsSIZE of the interior cells of each block given by $blockCellsSIZE = nZones(1) * nZones(2) * nZones(3)$ in case of $ndim = 3$. The blockCells contain the density of each cell element. FLASH, receiving the actual red-shift, the mass distribution on the mesh and the next red-shift of FLY, will be able to compute the hydrodynamic of the gas taking into account the DM distribution computed by FLY. FLASH evolves the system up to the next red-shift of FLY, the FLASH red-shift evolution being slower than the FLY red-shift evolution. After this communication phase FLY updates the system and starts the new time-step.

5. Conclusion

The software interface between FLY and FLASH, will allow us the execution of joint runs of FLY with hydrodynamical codes adopting the PARAMESH format.
The FLY interface may be adopted by any hydrodynamical code that uses the PARAMESH structure and we provide a simulator of the communication between FLY and a code sending the PARAMESH block structure.

References

Antonuccio, V. and Becciani, U. 2003, Comp. Phys. Comm., 155, 159

Fryxell , B. , et al. 2000, ApJ Supplement, 131, 273

Gropp, W. et al. 2005, http://www-unix.mcs.anl.gov/mpi/mpich2/index.htm#docs

MacNeice, P. 2000, Comp. Phys. Comm., 126, 330