METROC++

A parallel, object-oriented code for the computation of merger trees in cosmological simulations

Edoardo Carlesi
Leibniz Institut für Astrophysik
Potsdam, Germany
ecarlesi@aip.de

November 16, 2018

Contents

1 Introduction

METROC++ is an acronym that stands for **ME**rger**TR**ees **O**n **C++**, and refers to the infamous **Metro C** subway line in Rome, where the author of the code grew up and lived for more than 20 years. This legendary piece of infrastructure (which is still - in 2018 - largely under construction, though a shorter section of it is already operating) took something of the order of a few Giga-Years to be build, a time span which can be compared to the formation time of most dark matter halos.

The code has been written in C++ and relies on MPI2.0 C-bindings for the parallelization.

In addition to the source code, a number of bash and python scripts are provided to ease the

The properties of the algorithms used by the code are described in the code paper (Carlesi, MNRAS, 2018).

2 Basic usage

2.1 Compiling the code

The main folder contains a Makefile.config file

2.2 Code operation modes

2.3 Configuration file

2.4 Temporary files

The code produces a number of temporary files (.tmp format extension, located in the tmp/ folder). These files are needed at runtime to They can be produced manually or using the scripts (find_z.sh, find_n.sh located in the scripts/ folder) and contain the list of files on which the halo finder will run on. This can be the full list of snapshots in one simulation, or can be edited to be only a subset of it.

3 Examples

3.1 Full box simulation

3.2 Zoom simulation

4 Editing the code

4.1 Code structure

- `main.cpp` A wrapper for the functions determined elsewhere
- `utils.cpp` General functions and utilities are implemented here
- `spline.cpp` The spline class is used for interpolation
- `global_vars.cpp` A list of global variables accessible throughout the whole program
- `MergerTree.cpp` This file contains the merger tree class, that tracks the merging of the halos, and series of functions (FindProgenitors) that compute the trees themselves
- `IOSetting.cpp` Input/Output settings
- `Cosmology.cpp` Everything related to cosmological calculations
- `Communication.cpp` Handles most of the communication among tasks - sending / receiving buffers and so on
- `Grid.cpp` This handles the grid on which halos are placed and the buffers
- `Halo.cpp` The halo class contains the main halo properties

4.2 Other halo finders

Although `METROC++` was conceived and mainly tested using the `AHF` halo finder, it can be easily extended to support other software as well, as long as:

- Halo catalogues include informations about the number and types of particles, positions and velocities for each object
- Particle catalogs contain the (unique) IDs for each halo particles' content

To add

4.3 Improving the algorithms