XVIII International Conference on Water Resources
CMWR 2010
J. Carrera (Ed)
CIMNE, Barcelona 2010

AUTO-OPTIMIZATION ON PARALLEL HYDRODYNAMIC CODES: AN EXAMPLE OF COHERENS WITH OPENMP FOR MULTICORE

Francisco López-Castejón∗ and Domingo Giménez†

∗ Polytechnic University of Cartagena, Spain
e-mail: francisco.lopez@upct.es

† University of Murcia, Spain
e-mail: domingo@um.es, web page: http://www.um.es/pcgum/

Summary.

The simulation of marine scenarios is made through hydrodynamical models. These models are simulated with a high computational cost, and to reduce the simulation times parallel algorithms and systems have been used in recent years.

The development of efficient parallel algorithms is not easy, and even when the parallel software has been developed it is still not easy to obtain the best performance. The users of that software are normally non-experts in parallel programming and, furthermore, the optimal running conditions vary for different parallel computational systems.

With such a complex scenario, it is preferable to develop the software in such a way that it adapts automatically to the conditions of the computational system, so the user of the software does not need to know in detail the computational characteristics of the system to carry out the simulations in a reduced execution time.

We show how auto-optimization techniques can be included in the COHERENS marine simulation software. The methodology can be adapted to other softwares that work in a similar way. Multicore nodes are at present the basic components of the majority of the computational systems (from laptops and PCs to supercomputers, including networks of processors and clusters). So, the parallelization and auto-optimization have been done in OpenMP for multicore systems.

The simulations consist of a number of iterations where the values in two or three dimensional meshes are updated taking into consideration the values of some neighbouring points and some simulation conditions (salinity, pressure...) The software is structured in a number of loops, and each loop is paralleled independently. To obtain an autotunning version of the parallel program, each loop is paralleled with a parameter which determines the number of threads to be used. At running time the number of threads (a different number for each loop) is determined depending on the problem size and characteristics and on the computational system where the simulation is being carried out. To do that, some small

experiments are carried out when the software is installed in the system, and the decisions are taken at running time from the information generated by these experiments.

To validate the methodology, experiments have been carried out for different simulation conditions and in multicore nodes with different architectures with a number of cores from 2 to 16.