A MASSIVELY PARALLEL APPROACH TO THE QUASICLASSICAL REACTIVE SCATTERING

R. BARAGLIA, R. FERRINI, D. LAFORENZA, R. PEREGO
CNUCE, Italian National Research Council, Via S. Maria 36, 56126 Pisa, Italy

A. LAGANÀ
Dip. di Chimica, Università di Perugia, Via Elce di Sotto 8, 06100 Perugia, Italy

and

O. GERVASII
Centro di Calcolo, Università di Perugia, Piazza dell'Università 2, 06100 Perugia, Italy

Abstract

The suitability of massively parallel architectures for carrying out efficient calculations of quasiclassical rate constants for atom–diatom reactive processes has been investigated. Problems related to the parallel structuring of the computational procedure, fixed and scaled speedups, efficiency factors and their dependence upon the size of the problem, and the number of processors are discussed.

1. Introduction

Numerical solutions of scientific problems quite often rely on the use of intensive computing procedures. This means that to obtain these solutions in a realistic time, use of modern parallel features of advanced computers has to be made. To this purpose, both the numerical approach and the computing strategy need to be properly designed to take advantage of the parallelism.

The scientific problem considered in this paper is the calculation of rate constants for a family of elementary gas phase reactions relevant to the modeling of complex non-equilibrium systems starting from first principles. In particular, we have investigated bi-molecular atom–diatom reactions. These systems are interesting per se because important for several modern technological applications [1] and as a model for some larger molecule reactions.

The detailed state to state rate constant $k_{v_{ij}, v'_{ij'}}(T)$ at a given temperature $T$ can be evaluated by integrating over the collision energy ($E_{tr}$), the related reactive cross section $S_{v_{ij} v'_{ij'}}^{v_{ij} v'_{ij'}}(E_{tr})$, where $v_{ij}$ ($v'_{ij'}$) are the initial (final) vibrotational quantum numbers. In a quasiclassical mechanics approach, $S_{v_{ij} v'_{ij'}}^{v_{ij} v'_{ij'}}(E_{tr})$ can be formulated in terms of $P_{v_{ij} v'_{ij'}}(E_{tr})$, the atom–diatom reactive probability. Following a Monte Carlo approach [2], $P_{v_{ij} v'_{ij'}}(E_{tr})$ can be estimated from a limited number of trajectories $N$, by calculating the integral.
\[ P_{v_j, v_j'}(E_{tr}) = N^{-1} \int_0^1 d\xi_1 \int_0^1 d\xi_2 \int_0^1 d\xi_3 \int_0^1 d\xi_4 \int_0^1 d\xi_5 f_{v_j, v_j'}(\xi_1, \xi_2, \xi_3, \xi_4, \xi_5), \quad (1.1) \]

where the variables $\xi$ are related to initial position and momenta of the collision partners. The integrand $f_{v_j, v_j'}$ is a Boolean function that is unity when the integer value of $j'' + 1/2$ and $v'' + 1/2$ obtained from the trajectory calculation are, respectively, equal to $j'$ and $v'$ ($j''$ and $v''$ are the classical equivalent of the rotational and vibrational quantum numbers evaluated by integrating the classical equations of motion).

Each trajectory being a fully independent calculation, the integration of a batch of trajectories can be carried out in parallel by a pool of asynchronous processes cooperating according to a task farm model. A task farm model is based on a master process that manages the computation and dispatches the work to a set of slave processes. In a hypercube architecture, this cooperation model can be implemented by mapping the master process on the host node and the slave processes on the distributed nodes.

The paper is organized as follows: in section 2, the used hardware (NCUBE machines) and system software are given as well as the structuring of the program to be run on the parallel architecture are discussed. In section 3, speedups and performances of the restructured code are analysed.

2. Restructuring the code for a hypercube

A typical trajectory program consists of a preliminary section in which the random sequence initiator as well as physical constants are read in to generate some quantities of common use. The largest section of the program is embodied into a loop running over the trajectory index. Inside the loop, the necessary set of pseudo-random numbers are generated to obtain the starting conditions for the considered trajectory. After the validation of initial conditions, the recursive process of integrating in time positions and momenta starts. At the trajectory ending point, a determination of its final properties is performed to derive the products' properties and update the statistical analysis.

As already mentioned, to structure the program for the parallel environment \([3-6]\) we have adopted a task farm model of cooperation \([7]\) consisting of a master process running on the host and a set of identical worker processes loaded on the nodes. The master process (see scheme 1) takes care of all I/O operations, dispatches the trajectory calculations to the nodes, and collects final results. To optimize the load balancing, a self-scheduling method assigning only one trajectory at a time to each worker node has been adopted. An attempt has also been made to determine the number of trajectories assigned at a time to a worker process (granularity) which maximizes the load balancing while minimizing the communication overhead.