Experiences in Porting Applications to Cell/BE via CellSs

Annika Schiller and Godehard Sutmann

Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre, Research Centre Jülich, Germany
{a.schiller,g.sutmann}@fz-juelich.de

The Cell Broadband Engine (Cell/BE) is a heterogeneous multicore processor and has been designed to support a broad range of applications. It combines considerable floating-point resources required for demanding numerical algorithms with a power-efficient software-controlled memory hierarchy. When opening a new hardware architecture to a broad range of applications it is mandatory to enable the programming of applications running on Cell/BE for people who are no programming specialists. To simplify the enabling of applications on the Cell/BE, high-level programming models like Cell Superscalar Framework (CellSs) were developed. CellSs, developed and enhanced by Barcelona Supercomputing Centre (BSC), basically offers a high-level portable programming model to port, parallelize and tune applications on Cell/BE. The main advantages of CellSs compared to native programming are that (1) a complete rewriting of the application is not necessary, (2) it produces portable code and (3) many process stages are automatically invoked by the CellSs compiler, for example DMA transfer or distribution of work.

Via implementation of a triple-matrix-multiply (TMM) and a particle-based algorithm called Stochastic Rotation Dynamics (SRD) ease of programming and resulting performance of CellSs are assessed. The TMM realizes a Wavelet transform which is one part in the kernel of a program used to perform a Wavelet based evaluation of Coulomb potentials in molecular systems. The following TMM is considered

$$\tilde{A} \;= \mathcal{W}\, A\, \mathcal{W}^T \;\;\Rightarrow\;\; \tilde{A}_{ij} = \sum_{k,l} \mathcal{W}_{ik} \cdot A_{kl} \cdot \mathcal{W}_{jl} \tag{1}$$

where Wavelet matrix $\mathcal{W}$ is sparse and has a special banded structure and matrix A is dense and symmetric. The TMM was ported to Cell/BE via CellSs and the results were compared to the results obtained by using the native programming API of the IBM Software Development Kit (Cell SDK). In this work, CellSs was found to provide programmers a flexible programming model with an adaptive parallelism level that provided acceptable performance and portable code, while considerably simplifying Cell/BE programming.

SRD is a particle-based method for the simulation of hydrodynamic properties of fluid and flow phenomena on the mesoscale. The fluid is modeled by particles whose positions and velocities are continuous variables but the system evolves in discrete time steps. The evolution of the system consists of two basic steps: (1) a streaming step where the particles perform a uniform, linear motion without interacting with each other and (2) a collision step where multi-particle collisions are performed by a simultaneous stochastic rotation of the relative velocities of the particles. During the porting of the SRD algorithm via CellSs, several problems concerning load balancing, implementation of CellSs and diskless QS22 cluster nodes were encountered. The parallel program parts only provide an acceptable speedup when using two SPEs. When using more than two SPEs the distribution of work among the used SPEs is very imbalanced and idling times increase rapidly. The bad load balancing enormously limits the performance on the Cell/BE so that at current status, the calculation on Cell/BE is not faster than on usual PowerPC architectures.

It seems reasonable to conclude that at its current status CellSs is of limited use for complex applications depending on complex data structures. There is need for further development and enhancement especially with regard to usability to make debugging easier. As the programmer has no insight into the internal processes, a successful debugging is not possible. It would be desirable to provide more transparency so that internal processes can be followed and the construction of data dependencies can be understood. With the appearance of such kind of problems, the programmer is in a weak position and the development of efficient Cell/BE implementations becomes a challenging task, which is definitely not the aim of high-level programming languages.