8th. World Congress on Computational Mechanics (WCCM8)
5th European Congress on Computational Methods in Applied Sciences and Engineeering (ECCOMAS 2008)
June 30 –July 4, 2008
Venice, Italy

MPI SIMULATION OF DYNAMICS OF FLAMES

*** V. Karlin**

University of Central Lancashire
Preston PR1 2HE, United Kingdom
VKarlin@uclan.ac.uk

Key Words: *HPC performance, Global interprocessor communications, Premixed spherical flames.*

ABSTRACT

In this work we investigate effect of the MPI implementations of the cross-processor transposition of distributed data arrays on efficiency of multidimensional FFT based codes on modern HPC systems.

The code in question simulates dynamics of expansion of premixed spherical flames. The possibility of a self-induced transition of this type of flames to detonation is the issue of our interest in the applied side of this research. The governing equation is an asymptotic model of the Sivashinsky type and a spectral numerical algorithm is used to solve it. As a result the code relies heavily on global communications implementing interprocessor transposition of the global data array in which the numerical solution to the problem is stored. This global data interdependence makes interprocessor connectivity of the HPC system as important as the floating-point power of the processors of which the system is built. Details of the physical problem and numerical integration algorithm can be found in [1,2] and typical results of numerical simulations are illustrated in Figure 1.

Initially, the main computational array Ψ_{k_1,k_2} of size $K \times K$ is distributed between N_p processors column-wise, $K_p = K/N_p$ columns per processor. At this stage, the one-dimensional FFT can be effectively applied along the first index of the array, i.e. column-wise. In the next stage, the resulting global array should be transposed in order to accomplish the two-dimensional FFT by applying the one-dimensional FFT along the second index, i.e. raw-wise. The transposition of the global array requires cross-processor transmission of $K^2(1 - N_p^{-1})$ elements of Ψ_{k_1,k_2} and is the most communication resource-demanding segment of the code. Combined use of the FFT routine requires $O(K^2 \log_2 K)$ floating-point operations per time step making it the most crucial arithmetic resource consumer.

There are two straightforward strategies of implementing array transposition in MPI. First, the `MPI_GATHERV` routine can be called by every processor in order to fetch missing $(N_p - 1)K_pK/N_p$ elements of Ψ_{k_1,k_2} and form its raw-wise representation. Alternatively, the `MPI_ALLTOALL` routine can be applied to properly reshaped column-wise sub-arrays of Ψ_{k_1,k_2}. The reshaping is needed because the `MPI_ALLTOALL` routine requires the data destined for a particular processor to be stored in RAM continuously. Upon the transmission, the newly assembled sub-arrays should be reshaped back, now in the raw-wise manner, in order to apply the one-dimensional FFT along the second index. The sub-array reshaping trebles required memory, but allows it to use the maximum available communication bandwidth. As RAM is usually less critical to the problem in question than the interprocessor communications efficiency, the latter approach looks advantageous on HPC's with broader bandwidths.

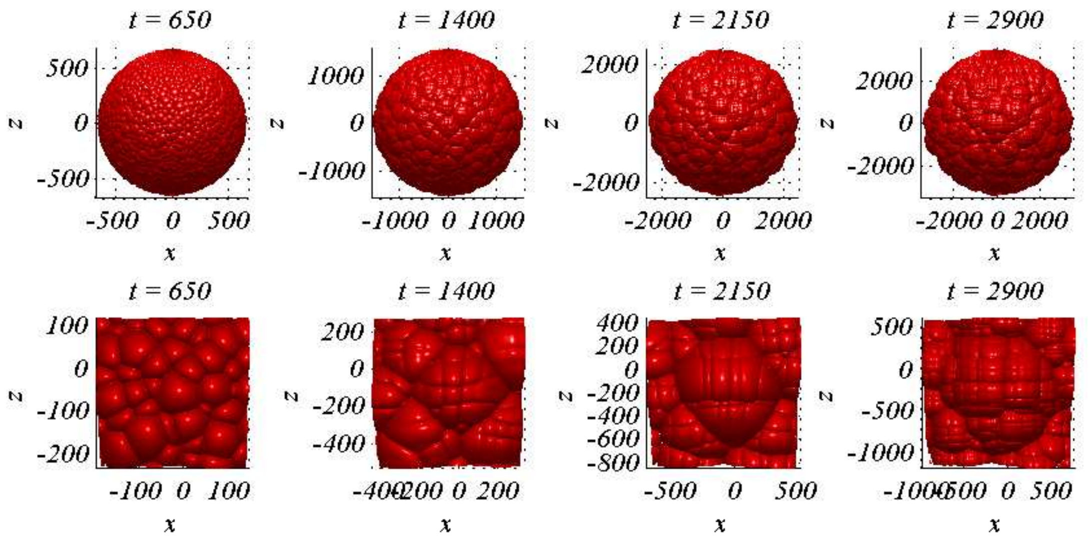

Figure 1: Evolution of half of a spherical flame (top) and of its central region (bottom).

The Fortran-90/MPI code underwent some basic optimization during porting on a particular HPC system. The NEC SX-8 required the largest optimization effort, though the pay off was the most ample too. Such basically optimized clones of the code were run on a set of HPC's with varying numbers of processors and solution array sizes.

All tested HPC systems favour the transposition based on the `MPI_ALLTOALL` routine, though the SGI Altix is faster with it just slightly and for relatively small arrays only. On average, our code runs on the SX-8 about 20 times faster than on the Altix. Further, it was found that the appropriately optimized code is perfectly scalable on the SX-8 in spite of its global all-to-all communications. In contrast, efficiency of the code drops rather rapidly on the Altix as the number of processors and/or size of the transposable array is increased. Profiling shows that further improvement of the code performance on all tested systems is impeded by the `MPI_ALLTOALL` routine and to a lesser degree by the FFT procedures.

Results of comparison of performances of our particular code on a set of HPC's should not be interpreted as the overall superiority of one of those HPC's over others.

ACKNOWLEDGEMENTS

This work was carried out under the HPC-EUROPA project (RII3-CT-2003-506079), with the support of the European Community - Research Infrastructure Action under the FP6 "Structuring the European Research Area" Programme. Support from the UK Consortium On Computational Combustion For Engineering Applications (COCCFEA) under the EPSRC grant No. EP/D080223/1 and from the HPCF-UCLan at the University of Central Lancashire is also appreciated.

REFERENCES

[1] V. Karlin and G. Sivashinsky. *Combust. Theory Model.*, Vol. **10**, 625–637, 2006.

[2] V. Karlin and G. Sivashinsky. *Proc. Combust. Inst.*, Vol. **31**, 1023–1030, 2007.