Programs description

February 11, 2017

This program is an MPI parallelized version of `compute-vacf.c` in `VACF-C` repository that computes the total velocity autocorrelation function (VACF) from LAMMPS [1] dump file (`compute_vacf.c`). Details of the program and VACF computations are described in the document accompanying the serial program and are unchanged.

MPI implementation distributes the main loop in the program among an arbitrary number of processors. The processors all read from the same file, compute VACF components for their atoms, and at the end send their local VACF to processor 0. Sending uses an MPI function `MPI_Reduce` which besides sending performs element addition on processor 0. To enable this communication in an efficient form local VACF array was transposed compared to the serial program allowing it's x, y, and z components to be contiguous in memory. After receiving the summed array, processor 0 performs the division by number of atoms and writes the result. Processor 0 also initially reads the number of atoms, frames, and the time array from the dump file and broadcasts it to all other processors using `MPI_Bcast`.

The program was benchmarked and ran for a 186 MB dump file on 1-32 processors and the results are shown in Figures 1-3. Values are reported in minutes and are an average of three measurements for each processor number. Speed up is defined as:

$$S = \frac{t_1}{t_p} \tag{1}$$

where t_1 is the execution time on 1 processor and t_p on p processors. Similarly, efficiency is

$$E = \frac{t_1}{p \times t_p} \times 100\% \tag{2}$$

Benchmarking was done with Intel Xeon CPU E5-2670 2.60GHz on two nodes of an HPC cluster. The cluster submission script is included. It also contains a comment on compilation of the program on that specific system. Benchmarking indicates good speed up and efficiency, which was expected due to lack of interprocessor communication except at the end of the program. Speed is most likely limited by collective access to the file and data transfer from the file. The file access speed is unclear as it is located on a common cluster partition with unknown connection type to particular nodes.

[1] S. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, J Comp Phys, 117, 1-19 (1995) (`http://lammps.sandia.gov/`)

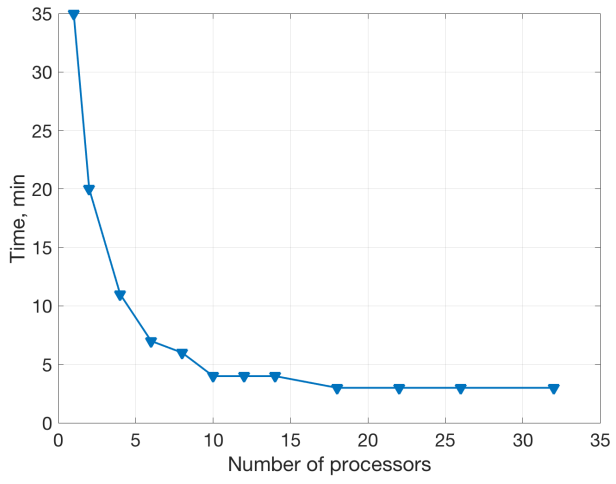

Figure 1: Execution time with varying processor number.

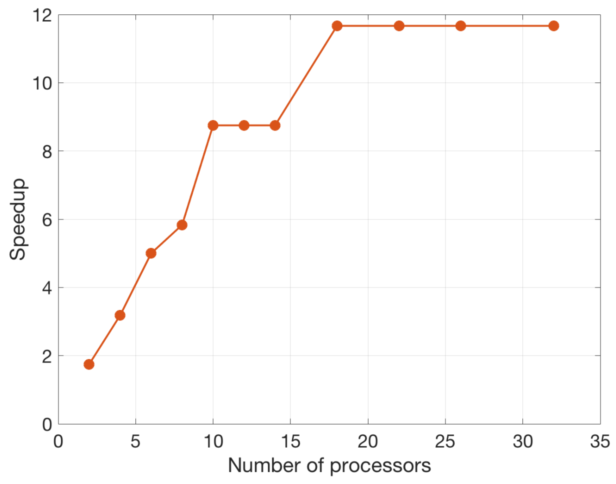

Figure 2: Speedup with varying processor number.

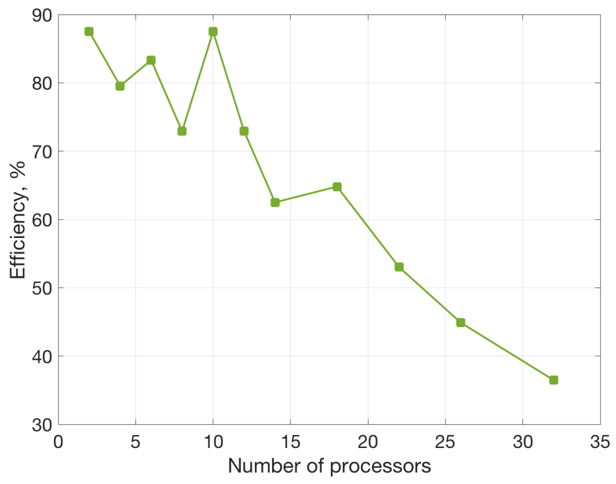

Figure 3: Efficiency with varying processor number.