A6: HPL Benchmark

Francesca Cairoli

Introduction

The aim of this exercise is to compile the HPL benchmark against the MKL multithread library and to tune it in order to reach at least the 75% of the theoretical peak performance of a node of the Ulysses cluster.

Benchmarks are used to assess the performance of machine by running a given program on it. Evaluating the computational complexity of a problem is therefore an important ingredient, but it is not always an easy task for real problems. The High Performance Linpack benchmark solves a dense, randomly generated, system of linear equations. The number of floating point operations needed to solve this problem is $2N^3/3 + 2N^2$, where N is the size of the system. This benchmark is used to determine the TOP500 list, which is a ranking of the most powerful computers. HPL is an accessible and portable implementation of the Linpack benchmark. Its parameters must be tuned in order to achieve high performances.

HPL benchmark

We downloaded the HPL software from `http://www.netlib.org/benchmark/hpl/hpl-2.2.tar.gz`. We followed the instructions provided in the assignment in modifying some compilation setting in the Make.mkl file and finally we compiled it.

The HPL parameters are: N, the size of the system, Nb, the block size, P and Q, indexes of rows and columns of a grid of MPI processes. We must look for a good combination of such parameters in order to achieve a good performance, hopefully greater than 75% of the TPP.

We are considering the following combinations: $N = 64512, 65000$, $Nb = 128, 192, 256$, $(P, Q) = (4, 5), (5, 4)$. Results are shown in Table 1. All the results are well above the 75% of the TPP, however the best results were reached for $N = 65000$, $Nb = 192$ and $(P, Q) = (4, 5)$, were we reached a peak performance of 418.1 GFlops, corresponding to the 93.32% of the theoretical peak performance.

We try the same parameters on the Intel, highly optimized, version of the Linpack benchmark. Results are shown in Table 2. As expected the Intel version outperformed the MKL implementation. In fact, a peak performance of $441,65$ GFlops was reached, which correspond to the 98.58% of the TPP.

N	NB	PxQ	Performance	% of TPP
64512	128	4×5	399.3	89.13%
64512	192	4×5	409.4	91.38%
64512	256	4×5	404.0	90.18%
64512	128	5×4	403.8	90.13%
64512	192	5×4	405.0	90.40%
64512	256	5×4	398.5	88.95%
65000	128	4×5	416.2	92.90%
65000	**192**	$\mathbf{4 \times 5}$	**418.1**	**93.32**%
65000	256	4×5	414.8	92.59%
65000	128	5×4	414.0	92.41%
65000	192	5×4	415.6	92.77%
65000	256	5×4	412.9	92.16%

Table 1: HPL+MKL results. The peak performance is measured in GFlops.

N	NB	Performance	% of TPP
64512	4×5	$433,75$	96.82%
65000	4×5	$441,65$	98.58%

Table 2: Intel benchmark average results. The peak performance is measured in GFlops.

Multi-process and multi-thread

The final request was to apply the HPL benchmark while varying the number of processors and threads, using the `OMP_NUM_THREADS` command, while keeping the size of the problem constant. The parameters N and Nb were set to best performing ones, while P and Q must change together with the number of MPI processes. In particular, the product of P and Q must coincide with the number of MPI processes. Results are summarized in the Table 3. The measured peak performance decreases with the number of processors even if the number of threads increases. Therefore, it seems like the varying number of threads is not affecting the results. However, using the `top -H` command inside the running node, we observed that 20 threads were activated, so the command `OMP_NUM_THREADS` has been recognized.

MPI proc	OMP threads	N	NB	PxQ	Performance	% of RPP
20	1	65000	192	4×5	418.1	93.33%
10	2	65000	192	2×5	226.5	50.56%
5	4	65000	192	1×5	124.7	27.83%
4	5	65000	192	2×2	102.3	22.83%
2	10	65000	192	1×2	51.10	11.41%
1	20	65000	192	1×1	26.91	6.01%

Table 3: Results of the multi-process and multi-thread HPL benchmark.