Homework 3

Question 2

*In this problem,* *you will modify the matmul.c program provided, optimizing the execution of the matrix multiplication with first a dense matrix, and second with a sparse matrix. You are welcome to use pthreads, OpenMP or any of the optimizations that were presented in class to accelerate this code. There will be prizes awarded for the fastest dense and the fastest sparse implementations.*

Dense multiplication

The code corresponding to the optimization of the dense matrix multiplication is provided in the file “Q2.c”. Two implementations have been studied for this case: blocked sequential matrix multiplication and parallel regular matrix multiplication. Table 1 reports the obtained results for both implementations. As it can be observed, the speedup granted by parallelism is much larger than the one obtained by blocking. Ideally, an implementation involving the two techniques will be better than each of them on its own. However, this combined implementation has not been explored in this report.

The threaded multiplication basically assigns the computation of a set of rows of the resulting matrix to each thread. To decide the number of threads to use for the parallel dense matrix multiplication, we run the code with different numbers of threads and evaluated the performance. The results are summarized in Figure 1, with 256 threads reporting the minimum computing time on a node with 32 CPUs (details in Figure 3).

Figure 1: Computing time vs. number of threads for the parallel dense matrix multiplication.

A similar test was performed to obtain the best block size to use in the blocked implementation. Results are reported in Figure 2. From the figure, lowest computing times are obtained with a block size of 32x32 elements, which is the one used in the comparison.

Figure 2: Computing time vs. block size (side of a square block).

Sparse matrix multiplication

The code corresponding to the optimization of the sparse matrix multiplication is provided in the file “Q2\_sparse.cpp”. For this operation we leveraged an open-source library called “Eigen”. This library has an API specific for sparse linear algebra, making it very simple for the programmer to implement the sparse matrix multiplication. Documentation can be found [here](https://eigen.tuxfamily.org/dox/). Results of the optimization are also included in Table 1.

The system used to obtain the results from Figure 1 and Table 1 has been a node of the Discovery cluster with the hardware characteristics listed in Figure 3.

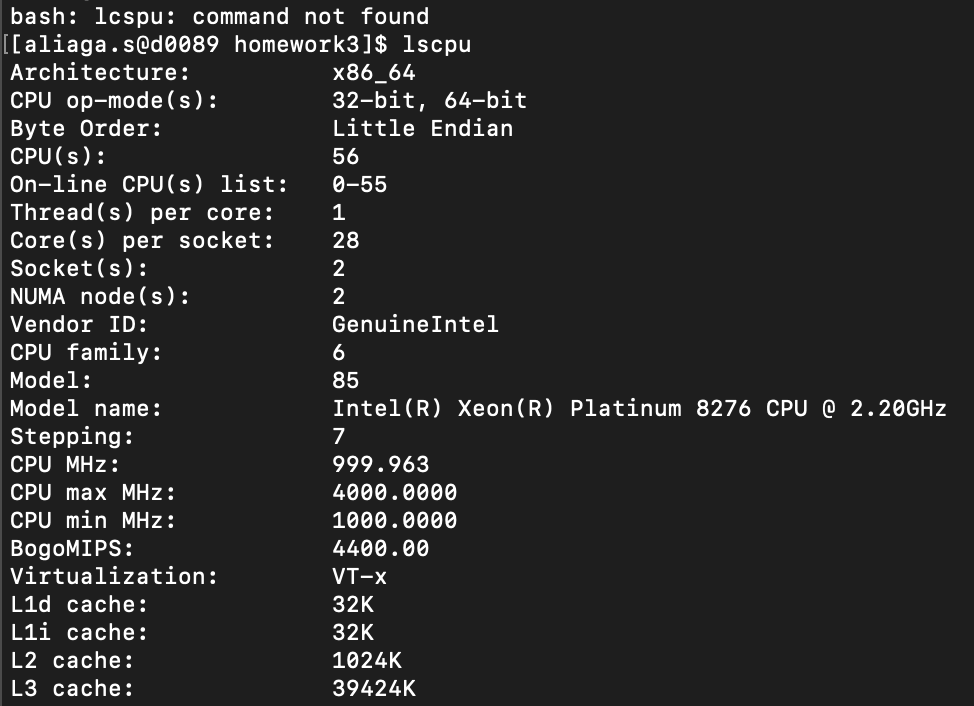


Figure 3: Characteristics of the hardware employed to run the tests presented in Table 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type of multiplication | **Regular** **(s)** (“matmul.c”) | **Optimized (s)**  (“Q2.c”/ “Q2\_sparse.cpp”) | | **Speedup** | |
| Blocked (32 block size) | Parallel (256 threads) |
| Dense | 8.0 | 3.2 | 0.41 | **2.5** | **19.5** |
| Sparse | 7.9 | 0.5 | | **15.8** | |

Table 1: Performance comparison of the different matrix multiplication implementations

From the results in Table 1 we can say that a great speed up with respect to the original implementation has been obtained. One thing to note is that the speedup obtained by using a pthread parallel implementation for the dense multiplication is greater than the one obtained for the sparse multiplication with the Eigen library. In the light of this result, we could have used the same implementation for the sparse multiplication but we believe that adding the results with Eigen justify even more the implementation with pthread.