CS60026: Parallel and Distributed Algorithms

Autumn 2016-17

Programming Assignment-1

Abstract. This assignment is aimed to familiarize you with OpenMP programming and for you to try out the various programming constructs taught in the class to see their effect on the execution time in multi-core environments. The problem that you are expected to solve in this assignment is *embarrassingly* parallel, that is, it is very obviously amenable to parallelization. At the same time, it provides a useful insight into the paradigms of hands-on parallel programming that a developer should keep in mind.

1 Sparse Matrices

In numerical analysis, a *sparse* matrix is a matrix in which most of the elements are zero. By contrast, if most of the elements are nonzero, then the matrix is considered *dense*. The number of zero-valued elements divided by the total number of elements ($m \times n$ for a $m \times n$ matrix) is called the *sparsity* of the matrix (which is essentially equal to 1 minus the density of the matrix). The concept of sparsity is useful in several engineering applications, which makes them relevant to us.

When storing and manipulating sparse matrices on a computer, it is beneficial and often necessary to use specialized algorithms and data structures that take advantage of the sparse structure of the matrix. Operations using standard dense-matrix structures and algorithms are slow and inefficient when applied to large sparse matrices as processing and memory are wasted on the zeros. Sparse data is by nature more easily compressed and thus require significantly less storage. Some very large sparse matrices are infeasible to manipulate using standard dense-matrix algorithms.

2 Data Structure for Sparse Matrices

In this assignment, we assume $n \times n$ square matrices that are stored in a *row-compressed* fashion. A description of row-compressed sparse matrices is provided below. Note that an analogous storage in a *column-compressed* fashion may be easily realized.

A row-compressed $n \times n$ sparse matrix consists of **three** arrays :

1. **`rowPtr`**: A Row Pointer is an integer array of size $n + 1$
2. **`colInd`**: A Column Index is an integer array of size **`numNonZeros`**, where **`numNonZeros`** is the total number of non-zero entries in the matrix

3. **`nnz`**: Non-Zeros is integer/float/double array of size **`numNonZeros`**

The Row Pointer and Columns Index arrays define the structure of the sparse matrix, while the Non-Zeros array contains the numerical entries.

The non-zeros in row i of the matrix can be accessed as follows :

- Find the starting column index **`colStart`** = **`rowPtr[i]`** and the last column index **`colEnd`** = **`rowPtr[i+1]-1`**
- For every **`j`** in **`[colStart, colEnd]`**, **`nnz[j]`** is a non-zero entry at row **`i`** and column **`colInd[j]`**

3 Matrix-Vector Multiplication

Multiplication of an $n \times n$ matrix A with an $n \times 1$ vector x to get the vector y can be described in terms of n dot-products of vectors. For $1 \leq i \leq n$, let $r_i = A(i,:)$, the i^{th} row of A. Then $y_i = \text{dot-product}(r_i, x)$.

4 Your Tasks:

The assignment folder contains an incomplete program *spmv.cc* and a *matrices* directory with 4 different sparse matrices.

1. Implement the serial version of sparse matrix-vector multiply in the **`multiply(...)`** function.

2. Implement a parallel version of **`multiply(...)`** using OpenMP.

3. Run your program on the 4 matrices provided in the matrices directory of the assignment folder.

4. Try different scheduling methods and find the one which produces the best multi-threaded scaling. You are expected to run your program with $1, 2, \cdots, 24$ threads and observe the performance scaling with the number of threads.

5. Write a short report with scaling charts for the different scheduling methods tried by you. Explain the result with emphasis on why a particular scheduling method works better than the others. How is it related to the type of sparse matrices provided in this assignment? Would the scaling performance turn out to be different for some other type of sparse matrices?

5 Submission Details

You are expected to submit a tar ball comprising of the completed code and your report. Submission deadline and site will be intimated soon. Enjoy!