CAAM 520 – Homework 1

Due 3/5/21, 11:59pm

General remarks: Please submit your homework report (as a PDF file) along with any code to your personal CAAM 520 Git repository. Please provide a makefile that can be used to compile your code, and make sure that your code compiles and runs in the CAAM 520 virtual machine. You are encouraged to discuss the problems with other students, but you must submit your own report and code. You may *not* consult solutions from previous years.

Problem 1 *(CPUs and caches, 20 pts.)*

Consider the following pieces of code, which operate on arrays `a`, `b`, and `c` of `double` values:

```
// CODE A:
for (int j = 0; j < m; j++) {
  for (int i = 0; i < n; i++) {
    a[i] += b[i]*c[i];
  }
}
```

```
// CODE B:
for (int i = 0; i < n; i++) {
  for (int j = 0; j < m; j++) {
    a[i] += b[i]*c[i];
  }
}
```

Suppose that both codes are run on a computer with the following (simplified) specifications:

- Clock frequency: 3.8 GHz
- L1 cache: 64 KiB with a bandwidth of 1 TiB/s
- L2 and L3 caches: none
- Main memory: 128 GiB with a bandwidth of 40 GiB/s

Assume that two CPU cycles are required to complete the computation `a[i] += b[i]*c[i]` for a single index `i` *if* `a[i]`, `b[i]`, and `c[i]` all reside in the L1 cache.

a) For a vector size of $n = 1024$, would you expect any difference in the performance of code A and code B? Explain your answer. (5 pts.)

b) For a vector size of $n = 4096$, would you expect any difference in the performance of code A and code B? Explain your answer. (10 pts.)

c) For a vector size of $n = 4096$, would you expect any difference in the performance of code A and code B if the machine's main memory was replaced by faster memory with a bandwidth of 60 GiB/s? Explain your answer. (5 pts.)

Note: When solving this problem, you do not have to account for any CPU features other than the L1 cache mentioned in the problem description, i.e., disregard superscalar architecture, SIMD instructions, pipelining, hyperthreading, etc. Furthermore, disregard any effects that memory latencies, compiler optimizations, or the operating system might have on the results. Recall that a value of type double consumes 8 Bytes and that 1 TiB = 1024 GiB = 1024^2 MiB = 1024^3 KiB = 1024^4 Bytes.

Problem 2 *(Amdahl's law, 15 pts.)*

Consider an application that performs the total amount of work $s + p = 1$, where p is the fraction of the work that can be performed in parallel. Assume that the parallelized part of the application exhibits linear scalability, i.e., assume that

$$T_{f,p}^N = \frac{T_{f,p}^1}{N}$$

where N is the number of workers and the subscript f indicates that the total amount of work is fixed. For this particular application, the fraction of the work that must be done sequentially is 15%, i.e., $s = 0.15$.

a) Compute the speedup and parallel efficiency achieved when using $N = 4$ workers. (5 pts.)

b) Compute the limits of the speedup and parallel efficiency as the number of workers is increased indefinitely ($N \to \infty$). (5 pts.)

c) Compute the maximum number of workers for which a parallel efficiency of at least 50%, i.e., $E_N \geq 0.5$, can be achieved. (5 pts.)

Problem 3 *(Race conditions and thread safety, 15 pts.)*

Consider the following function, which increments a global counter and returns its new value:

a) Is the `increment()` function thread safe, i.e., does it still exhibit the desired behavior when called concurrently by multiple (OpenMP) threads? Explain your answer. (5 pts.)

b) If the function is not thread safe, modify the code to make it thread safe. (5 pts.)

c) Add a function `void reset()` which resets the value of the global counter variable to zero. Make sure that the `increment()` and `reset()` functions are still thread safe, keeping in mind that both functions access the *same* global variable. (5 pts.)

```
int counter; // global variable

int increment()
{
  counter++;
  return counter;
}
```

Note: Please submit the code for this problem as part of your report, not as a separate source code file.

Problem 4 *(Sparse matrices and the Jacobi method using OpenMP, 50 pts.)*

The Jacobi iteration

$$x^{(k+1)} = x^{(k)} + \omega D^{-1}(b - Ax^{(k)})$$

is one of the simplest iterative methods to approximate the solution of a linear system $Ax = b$. Here, $\omega > 0$ is a real number called the relaxation coefficient, and D is the diagonal part of A, i.e.,

$$D_{ij} = \begin{cases} A_{ij}, & \text{if} i = j, \\ 0, & \text{otherwise} \end{cases}.$$

The iteration starts with an initial guess $x^{(0)}$ and stops once the relative residual

$$\frac{\|b - Ax^{(k)}\|_2}{\|b\|_2} \tag{1}$$

is less than some tolerance $\varepsilon > 0$.

In this problem, you will implement the Jacobi method for a general linear system $Ax = b$, where A is a sparse matrix in the compressed row storage (CSR) format.

a) Familiarize yourself with the CSR format for sparse matrices by reviewing the example [2]. Be sure to understand the role of the three arrays `row_ptr`, `col_ind`, and `val`. (0 pts.)

b) Review the structure of the `csr_matrix_t` data type in `csr_matrix.h` as well as the `main()` and `jacobi()` functions in `jacobi.c`. (0 pts.)

c) Use OpenMP to implement a parallel Jacobi method in the `jacobi()` function in the file `jacobi.c` which is provided as part of this homework assignment. Do not change the `main()` function or the signature of the `jacobi()` function. Make sure that your implementation of the `jacobi()` function returns the number of iterations and the relative residual as defined in (1). (35 pts.)

Hint: Start by implementing a function that computes the sparse matrix-vector products $Ax^{(k)}$ in (1) and continue from there.

d) Test your implementation using the matrices below. For each test case, report the number of Jacobi iterations needed to achieve a tolerance of $\varepsilon = 10^{-1}$. (10 pts.)

- `bcsstk09` (file `bcsstk09.mtx`) with $\omega = 1.0$
- `bcsstk11` (file `bcsstk11.mtx`) with $\omega = 0.5$
- `bcsstm12` (file `bcsstm12.mtx`) with $\omega = 0.4$

Note: Simply click the links above to download each matrix from the SuiteSparse Matrix Collection [1].

e) Your implementation of the Jacobi method should contain a `for` loop over the rows of the sparse matrix A. Should OpenMP's static or dynamic schedule be used for this loop, or does the optimal choice depend on the matrix at hand? If so, which properties of A should influence your choice? (5 pts.)

Remark: The Jacobi method that you implemented in this problem only converges for system matrices A that satisfy $\rho(D^{-1}R) < 1$ [3, section 4.2.1]. Hence, it does not converge for general sparse linear systems, and when it does converge, the convergence rate is often unsatisfactory. Nonetheless, the Jacobi method is an important ingredient for multigrid methods [3, chapter 13], a class of highly efficient linear solvers.

References

[1] SuiteSparse matrix collection. `https://sparse.tamu.edu/`.

[2] Jack Dongarra. Compressed row storage (CRS). `http://netlib.org/linalg/html_templates/node91.html`, 1995.

[3] Yousef Saad. *Iterative methods for sparse linear systems.* SIAM, 2003.