MCS 572 Project Two : parallel tropical matrix multiplication

Shared Memory Parallel Tropical Matrix Multiplication

The goal of the project is to use shared memory parallelism to multiply two tropical matrices.

The tropical semiring is $\mathbb{R} \cup \{-\infty\}, \oplus, \otimes$, with $x \oplus y = \max(x, y)$ and $x \otimes y = x + y$. In this semiring $-\infty$ plays the role of zero for $\oplus$ as $x \oplus -\infty = \max(x, -\infty) = x$ and for $\otimes$ as $x \otimes -\infty = x - \infty = -\infty$. This max-plus algebra has applications in the control of discrete event systems, see `http://maxplus.org`.

In C, in `math.h`, the macro `INFINITY` is defined and we can compute with `-INFINITY` and doubles. The `-INFINITY` shows as `-inf` when given as argument of `printf()`.

Let A be an ℓ-by-m matrix and B be an m-by-n matrix, respectively with elements $a_{i,j}, b_{i,j} \in \mathbb{R} \cup \{-\infty\}$. The product of A with B is $C = A \odot B$, an ℓ-by-n matrix with elements $c_{i,j}$ defined as

$$c_{i,j} = \bigoplus_{k=1}^{m} a_{i,k} \otimes b_{k,j} = \max_{k=1}^{m}(a_{i,k} + b_{k,j}).$$

For example, consider a 3-by-5 matrix `A` multiplied with a 5-by-4 matrix `B` with elements `-inf`, `0`, and `1` positioned at random in `A` and `B`, their product is a 3-by-4 matrix `A*B` shown below.

```
The matrix A :
 -inf 0.00 -inf -inf 1.00
 1.00 -inf 1.00 1.00 -inf
 1.00 1.00 0.00 0.00 -inf
The matrix B :
 -inf -inf 0.00 0.00
 0.00 -inf 0.00 1.00
 -inf 0.00 1.00 0.00
 -inf 0.00 0.00 0.00
 1.00 -inf 0.00 -inf
The matrix A*B :
 2.00 -inf 1.00 1.00
 -inf 1.00 2.00 1.00
 1.00 0.00 1.00 2.00
```

This is the output of a simple C program posted at the course web site. The other, recommended language for this project is Julia. At the course web site there is a simple Julia program. We briefly touched on Numba and Parsl as tools for multithreading in Python and the course web site has a simple Python script which defines the tropical matrix multiplication.

Julia provides both threading and tasking constructs. In C, your parallel shared memory implementation may use OpenMP, pthreads, or the Threading Building Blocks. In Python, you may apply Numba or Parsl.

Assignment One: define the shared memory parallelism

Write code to compute the multiplication of two matrices, on a shared memory parallel computer. Your program should take five arguments at the command line:

1. ℓ: the number of rows of the first matrix;
2. m: the number of columns of the first matrix, which equals the number of rows of the second matrix;
3. n: the number of columns of the second matrix;
4. p: the number of threads; and
5. N: the times the parallel multiplication must be executed.

Run tests for sufficiently large matrices, each time generating a new pair of random matrices. Report the wall clock time and the elapsed CPU time to measure the speed ups. Take into account the time it takes to generate the matrices.

Indicate whether you used pascal or ampere, or provide the specifications of your computer.

Answer the following questions:

1. If multiplying A with B, the second matrix B is stored column wise instead of row wise, does the code run then faster?
2. Compare the cost of the tropical matrix-matrix multiplication with the cost of ordinary matrix-matrix multiplication. Is there a significant difference?

Assignment Two: discuss the granularity

Consider three types of granularity:

1. The coarsest granularity distributes the first ℓ-by-m matrix A among p threads row wise. The matrix A is partitioned into p strips, where each strip has n/p entire rows of A.
2. For a given block size b, the matrices A and B are structured as matrices of b-by-b matrices. The matrix-matrix multiplication is then executed in a tiled fashion.
3. Another granularity considers the matrix-matrix multiplication of an ℓ-by-m with an m-by-n matrix as the computation of $\ell \times n$ inner products. Each inner product leads to one element in the product and defines a different job in the parallel computation.

Compare the advantages and disadvantages of each type of granularity. In your discussion, distinguish between threading and tasking.

Assignment Three: examine the scalability

Answer the following questions:

1. For which dimensions gives a multithreaded matrix-matrix multiplication a significant speedup?
2. Relate the number of threads to the dimensions for which there is a significant speedup?
3. Can you derive an iso-efficiency analysis for this problem?
4. Draw the directed acyclic graph of the tasks in your parallel implementation.
   Do a critical path analysis to determine the maximum speedup that can be achieved.

Is this problem memory bound or compute bound? Refer to your answer to question 2 of Assignment One.

The deadline is Monday 28 October 2024 at noon

Submit your solution (code, output of runs, and report) to gradescope.

You may work individually or in pairs on this project. For each pair, please submit only one solution.

If you have questions or difficulties with the assignments, feel free to come to my office for help.