Παράλληλα και Διανεμημένα Συστήματα

Εργασία 4 [2020-2021] Υποβολή/Αξιολόγηση

Return to: Εργασίες και Τε...

Φάση υποβολής

Φάση εγκατάστασης	**Φάση υποβολής** Τρέχουσα φάση	Φάση αξιολόγησης	Φάση αποτίμησης βαθμολόγησης	Λήξη
	Υποβολή της εργασίας σας Ανοιχτό για υποβολές από Wednesday, 6 January 2021, 12:00 AM (23 ημέρες πριν) Καταληκτική ημερομηνία	Ανοιχτό για αξιολόγηση από Monday, 1 March 2021, 11:55 PM (32 μέρες ακόμα) Προθεσμία αξιολόγησης: Monday, 8 March 2021, 11:55 PM (39		

Οδηγίες για την υποβολή

Parallel & Distributed Computer Systems (Jan 25, 2021) Exercise 4

Sparse Boolean Matrix Multiplication with two levels of parallelization

In this assignment we will implement Boolean matrix multiplication (BMM) of sparse CSC matrices using at least two forms of parallelism that combined produce the correct result faster than either form of parallelism independently. You may combine shared and distributed memory parallelism either OpenMP or Cilk, together with MPI, and or GPU programming.

We denote the logical conjunction (and), disjunction (or) and negation (not) by $\wedge$, $\vee$ and $\neg$, respectively. Given matrix $A = A(I, K) = [a(i, k)]$ with row index set I and column index set K, matrix $B = B(K, J) = [b(k, j)]$ with index sets K and J. Let $n_i = |I|$, $n_j = |J|$ and $n_k = |K|$. Let $C = C(I, J) = [c(i, j)]$ be the Boolean product $C = F \odot (AB) \colon c(i, j) = \bigvee_{k \in K} a(i, k) \wedge b(k, j), \quad k \in K, \quad i \in I, \quad j \in J, \quad F(i, j) \neq 0.$

Element $c(i, j) = 1$ if and only if there exists k in the search domain K such that $a(i, k)b(k, j) = 1$, i.e., there is a match between the binary vector in row i of A and the binary vector in column j of B.

If provided a filter F on the product AB, then only those elements with $F(i, j) = 1$ are calculated. The filtered product can be expressed as $F \odot (AB)$, where $\odot$ denotes the Hadamard product, i.e., element-wise product. The absence of a prescribed filter is equivalent to the case where F has no zero elements. We may therefore consider general BMM in the so-called output-sensitive form $F \odot (AB)$.

Similar to algebraic matrix multiplication, BMM can be written in block version. Assume here for convenience that $n_I = n_J = n_K = n$. Let b be the block size. Suppose $n_b = n/b$ be an integer greater than b. Partition each index set into n_b subsets of equal size b. The bipartite matrices A, B and C are accordingly partitioned into $n_b \times n_b$ block matrices with blocks of size $b \times b$.

Denote by $C_{p,q}$ the (p, q) block of matrix C, $1 \le p, \quad q \le n_b$. The matrix blocks of A, B and F are similarly denoted. Then,

```
    $C_{p,q} := 0;$
    for $s := 1, \cdots, n_b$
        $C_{p,q} := C_{p,q} \vee F_{p,q} \odot (A_{p,s}\, B_{s,q})$
```

There are n_b^2 blocks in C, and there are n_b^3 block products $A_{p,s}B_{s,q}$ in total.

Your implementation does not need to be recursive. Select a block size b that provides best performance for matrices that have n more than a million.

You should provide two different interfaces, in the same way as the following two `MATLAB` routines:

```
% MATLAB uses ordinary multiplication with doubles
% To emulate BMM, we compare against 0 to get true/false values
% do not implement your code with doubles!
function C = bmm(A,B)
C = (A*B)>0;
end

function C = bmmfiltered(A,B,F)
C = ( F.*(A*B) )>0;
end
```

Before proceeding to any parallel implementation, verify that your sequential code is correct and its wall-clock execution time is comparable to the following MATLAB/Octave commands

```
n = 5e6;
d = 2; % approximate number of true elements per row

A = sprand( n, n, d/n ) > 0;
B = sprand( n, n, d/n ) > 0;

tic;
C = (A*B) > 0;
toc
```

Optional improvements

Output sensitive product

There are redundant operations in BMM by the naive calculation, especially when matrices A and B are dense. Once a `true` is computed in the accumulation for element $c(i, j)$, the rest of the accumulation is redundant and can be skipped. This also integrates with the prescribed filter F,

```
    $C_{p,q} := 0; \quad X_{p,q} := F_{p,q}$
    for $s := 1, \cdots, n_b$
        $C_{p,q} := C_{p,q} \vee F_{p,q} \odot (A_{p,s}\, B_{s,q})$
        $X_{p,q} := X_{p,q} \wedge (\neg C_{p,q})$.
```

The remaining block BMMs, during the accumulation, are adaptively filtered, i.e., made output sensitive.

Method of Four Russians

The block BMMs at the base level, $C_{p,q} := C_{p,q} \vee (X_{p,q} \odot (A_{p,s}B_{s,q}))$, may be accelerated by the Four Russians (4R) algorithm.

What to submit

- A 3-page report in PDF format (any pages after the 3rd one will not be taken into account).
  - Describe in a paragraph two applications of BMM in science and engineering in adequate detail.
  - Describe your parallel algorithm design main points, data distribution, communication patterns and blocking factor choice decisions.
  - Report execution times of your implementations with respect to size n on the sets of matrices we will specify. Use innovative and imformative plots to convey the information and explain the performance you get
- Upload the source code on GitHub, BitBucket, Dropbox, Google Drive, etc. and add a link in your report.
- Cite the sources of everything you used.
- You may work in groups of two. Submit report twice with both collaborator names listed.

Deadline

The weekend after the end of February or September exam period, (you may also submit in June, provided you are registered).

Η υποβολή σας
Δεν έχετε υποβάλει την εργασία σας ακόμα

Έναρξη προετοιμασίας της υποβολής σας

◄ [Εργασία 4 2020-2021] Επιλογή ομάδας

Βαθμοί Σεπτεμβρίου 2019 ►

Return to: Εργασίες και Τε...