CSL361 Problem set 6: Speeding up computations is an art and block matrix operations help

March 24, 2017

(For self study)

Block matrix algorithms are those which are "rich" in matrix-matrix multiplications. Algorithms of this type are likely to be more efficient than those that just manipulate scalars because more computation is involved with a given data movement making such algorithms more cache efficient. For example, k by k matrix multiplication involves $2k^3$ flops but only $2k^2$ data. In this tutorial we will investigate some basic block matrix operations.

Consider the following partitioning (blocking) of an $m \times n$ matrix A.

$$A = \begin{bmatrix} A_{11} & \dots & A_{1q} \\ \vdots & & \vdots \\ A_{p1} & \dots & A_{pq} \end{bmatrix} \begin{matrix} m_1 \\ \vdots \\ m_p \end{matrix}$$
$$\begin{matrix} n_1 & \dots & n_q \end{matrix}$$

where each A_{ij} has m_i rows and n_j columns such that $m_1 + \ldots + m_p = m$ and $n_1 + \ldots + n_q = n$. We say that A is a $p \times q$ block matrix. If a block matrix B has the same structure, we say that B is partitioned *conformably* with A.

1. Verify that if A and B are conformable block matrices then $C = A + B$ is also conformable with $C_{ij} = A_{ij} + B_{ij}$.

2. Suppose $A \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{r \times n}$ are partitioned as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{matrix} m_1 \\ m_2 \end{matrix} \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \begin{matrix} r_1 \\ r_2 \end{matrix}$$
$$\begin{matrix} r_1 & r_2 \end{matrix} \qquad \begin{matrix} n_1 & n_2 \end{matrix}$$

with A_{ij} of size $m_i \times r_j$ and B_{ij} of size $r_i \times n_j$, with $m_1 + m_2 = m$, $r_1 + r_2 = r$ and $n_1 + n_2 = n$. Show that

$$AB = C \;=\; \begin{array}{c} \left[\begin{array}{cc} C_{11} & C_{12} \\ C_{21} & C_{22} \end{array}\right] \\ \begin{array}{cc} n_1 & n_2 \end{array} \end{array} \begin{array}{c} m_1 \\ m_2 \end{array}$$

with C_{ij} of size $m_i \times n_j$ where $C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j}$ for $i = 1, 2$ and $j = 1, 2$.

3. Suppose

$$A \;=\; \begin{array}{c} \left[\begin{array}{ccc} A_{11} & \dots & A_{1q} \\ \vdots & & \vdots \\ A_{p1} & \dots & A_{pq} \end{array}\right] \\ \begin{array}{ccc} r_1 & \dots & r_q \end{array} \end{array} \begin{array}{c} m_1 \\ \vdots \\ m_p \end{array} \qquad B \;=\; \begin{array}{c} \left[\begin{array}{ccc} B_{11} & \dots & B_{1t} \\ \vdots & & \vdots \\ B_{q1} & \dots & B_{qt} \end{array}\right] \\ \begin{array}{ccc} n_1 & \dots & n_t \end{array} \end{array} \begin{array}{c} r_1 \\ \vdots \\ r_q \end{array}$$

where A_{ij} of size $m_i \times r_j$ and B_{ij} of size $r_i \times n_j$ with $m_1 + \ldots + m_p = m$, $r_1 + \ldots + r_q = r$ and $n_1 + \ldots + n_t = n$.

If we partition the product $C = AB$ as follows

$$C \;=\; \begin{array}{c} \left[\begin{array}{ccc} C_{11} & \dots & C_{1t} \\ \vdots & & \vdots \\ C_{p1} & \dots & C_{pt} \end{array}\right] \\ \begin{array}{ccc} n_1 & \dots & n_t \end{array} \end{array} \begin{array}{c} m_1 \\ \vdots \\ m_p \end{array}$$

with C_{ij} of size $m_i \times n_j$ then show that

$$C_{ij} = \sum_{k=1}^{q} A_{ik}B_{kj} \text{ for } i = 1 : p \text{ and } j = 1 : t$$

[**Hint:** Use induction along with the previous problem]

4. A special case of the above is **block matrix times vector**. Suppose we partition

$$A \;=\; \begin{array}{c} \left[\begin{array}{c} A_1 \\ \vdots \\ A_p \end{array}\right] \end{array} \begin{array}{c} m_1 \\ \vdots \\ m_p \end{array} \qquad z \;=\; \begin{array}{c} \left[\begin{array}{c} z_1 \\ \vdots \\ z_p \end{array}\right] \end{array} \begin{array}{c} m_1 \\ \vdots \\ m_p \end{array}$$

where each A_i and z_i has m_i rows and $m_1 + \ldots + m_p = m$. We refer to A_i as the i-th block row. If $x \in \mathbb{R}^n$ and $Ax = z$ then show that $A_i x = z_i$ for $i = 1 : p$. Also, show that each $z_i = A_i x$ can be computed using either **dot** or **saxpy** procedures.

5. Another way to partition the matrix-vector multiplication problem is to partition A and x as follows

$$A = \begin{bmatrix} A_1 & \dots & A_q \\ n_1 & \dots & n_q \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ \vdots \\ x_q \end{bmatrix} \begin{matrix} n_1 \\ \vdots \\ n_q \end{matrix}$$

where each A_i and x_i has n_i columns and rows respectively. We refer to A_i as the i-th block column. Show that the product $z = Ax$ has the form $z = A_1x_1 + \dots A_qx_q$. Also, show that each $z_i = A_ix_i$ can be computed using either **dot** or **saxpy** procedures.

6. Envisage situations where each of the above four ways of computing "matrix times vector" may be preferable.

7. Just as scalar level matrix multiplications can be arranged in several different ways, so can the multiplication of block matrices.

   (a) Assume (for simplicity) that $A = (A_{ij})$ and $B = (B_{ij})$ are each $N \times N$ block matrices with $\alpha \times \alpha$ blocks, i.e., $n = N\alpha$. With this uniform blocking, if $C = (C_{ij}) = AB$, then clearly

   $$C_{ij} = \sum_{k_b=1}^{N} A_{ik_b}B_{k_bj} \qquad i = 1:N \quad j = 1:N$$

   gives a **block dot product** procedure.

   (b) If we partition A into block columns

   $$A = \begin{bmatrix} A_1 & \dots & A_N \end{bmatrix}$$

   where each A_j has α columns, and if $B = (B_{ij})$ with $B_{ij} \in \mathbb{R}^{\alpha\times\alpha}$, then clearly

   $$AB = C = \begin{bmatrix} C_1 & \dots & C_N \end{bmatrix} \text{ with } C_j = \sum_{k=1}^{N} A_kB_{kj}$$

   gives a **block gaxpy** procedure.

   (c) If we partition A into block columns (of size α) as before

   $$A = \begin{bmatrix} A_1 & \dots & A_N \end{bmatrix}$$

and B into block rows (of size α)

$$B = \begin{bmatrix} B_1 \\ \vdots \\ B_N \end{bmatrix}$$

then clearly

$$C = AB = \sum_{k_b=1}^{N} A_{k_b} B_{k_b}$$

gives a **block outer product** procedure.

Try to envisage situations under which each may be preferable and develop procedures for the above. (Good luck!)

8. Presumably you are familiar with the store-by-row (row major) and store-by-column (column major) data structures for matrices (if not, please look up any book on data structures, or ask somebody suitable). Design a store-by-block data structure for matrices and show circumstances under which such a scheme may be useful.

9. Consider Problem 2. In the naive algorithm, $C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j}$, there are 8 multiplications and 4 additions. Strassen (V. Strassen (1969). "Gaussian Elimination is Not Optimal", *Numer. Math. 13*, 354-356) showed how to compute C with 7 multiplications and 18 additions:

$$\begin{aligned}
P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
P_2 &= (A_{21} + A_{22})B_{11} \\
P_3 &= A_{11}(B_{12} - B_{22}) \\
P_4 &= A_{22}(B_{21} - B_{11}) \\
P_5 &= (A_{11} + A_{12})B_{22} \\
P_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\
P_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \\
C_{11} &= P_1 + P_4 - P_5 + P_7 \\
C_{12} &= P_3 + P_5 \\
C_{21} &= P_2 + P_4 \\
C_{22} &= P_1 + P_3 - P_2 + P_6
\end{aligned}$$

Please confirm by substitution (I would strongly advise that you use a symbolic package; this may be a good opportunity to figure out how to use one). Suppose $n = 2m$ so that the blocks are $m \times m$. Counting

operations in $C = AB$ we find that conventional matrix multiplication involves $(2m)^3$ multiplications and $(2m)^3 - (2m)^2$ additions. In contrast, if Strassen's algorithm is applied *with conventional multiplication at the block level*, then $7m^3$ multiplications and $7m^3 + 11m^2$ additions are required. If $m >> 1$ then the Strassen's method involves about 7/8-th of the arithmetic of the conventional algorithm.

Of course (with our CSL201 training in *recursion* and *divide-and conquer*) we can recurse (a bit problematic if n is not a power of 2, but surely we can handle even that?). There is no need to recurse all the way down to $n = 1$; when the blocks are sufficiently small ($n \leq n_{min}$) it may be sensible to use conventional matrix multiplication.

(a) Write a recursive procedure for Strassen's method.

(b) If $n_{min} >> 1$ it suffices to count only the multiplications as the number of additions are roughly the same. Show that the number of multiplications required is $O(n^{\log_2 7}) = O(n^{2.807})$.

(c) Argue (as convincingly as possible) that Strassen's method is of dubious practical value.

(d) Argue (as convincingly as possible) that Strassen's method, if used judiciously and cleverly, can indeed be useful. In case you are not convinced, D. Bailey (D. Bailey (1988). "Extra High Speed Matrix Multiplication on the Cray-2", *SIAM J. Sci. and Stat. Comp.*, 9, 603-607) showed that with $n_{min} = 128$ his Strassen approach required about 60% time compared to conventional matrix multiplication.

10. Consider the n by n matrix multiplication $C = AB$. Assume that the cache can hold M floating point numbers and that $M << n^2$. Partition B and C into columns

$$B = \underset{\alpha \quad \ldots \quad \alpha}{\left[\begin{array}{ccc} B_1 & \ldots & B_N \end{array} \right]} \qquad C = \underset{\alpha \quad \ldots \quad \alpha}{\left[\begin{array}{ccc} C_1 & \ldots & C_N \end{array} \right]}$$

where $n = \alpha N$. Suppose α has been chosen such that $M \approx n(2\alpha+1) \approx 2n^2/N$ such that a block column of B, a block column of C, and a column of A can fit into the cache. Here is a matrix multiply procedure that gets a fair amount of re-use for each B_j brought into the cache.

for $j = 1 : N$
  Load B_j and $C_j = 0$ into cache

```
    for k = 1 : n
        Load A(:, k) into cache and update C_j
    end
    Store completed C_j in main memory
end
```

The k loop performs a **saxpy** oriented matrix product $C_j = AB_j$. Verify that the number of floating point numbers that move through the cache and main memory "door" during execution of the **block saxpy** algorithm is

$$\sum_{j=1}^{N}[n(n/N) + n^2 + n(n/N)] = (2+N)n^2 = 2n^2(1+n^2/M)$$

Conclude that a large cache helps.

11. A more effective cache utilization results with the following block dot product scheme. Regard $A = (A_{ij})$, $B = (B_{ij})$ and $C = (C_{ij})$ as N-by-N block matrices with uniform block size $\alpha = n/N$. Assume that N is chosen such that $3\alpha^2 \approx M$, i.e., 3 blocks can fit into the cache. With C_L being the cache workspace, we describe the procedure as follows:

```
for i = 1 : N
  for j = 1 : N
    C_L(1 : α, 1 : α) = 0
    for k = 1 : N
      Load A_ik and B_kj into cache
      C_L = C_L + A_ik B_kj
    end
    Store C_L in main memory location for C_ij
  end
end
```

Show that with this organization the main memory - cache traffic sums as

$$n^2(1+2N) \approx 2n^3\sqrt{3}/\sqrt{M}$$

If $n = 2^{10}$ and $M = 2^{14}$ by what factor do the traffic for the two methods differ?