Distributed Matrix Multiplication

Venkata Paredla
201551035

Introduction

- Multiplication between two matrices requires huge computational power and also requires more memory, when the size of matrix is large.
- When size of matrices become larger, it becomes inefficient and time consumption task.

Problem Definition

Given a matrix , A ($m \times n$) m rows and r columns, where each of its elements is denoted aij With $1 \le i \le m$ and $1 \le j \le r$, and a matrix B ($r \times n$) of r rows and n columns, Where each of its elements is denoted bij with $1 \le i \le r$, and $1 \le j \le n$, the matrix C resulting from the operation of multiplication of matrices A and B, $C = A \times B$, is such that each of its elements is denoted ij with $1 \le i \le m$.

Traditional Approach

- Calculated as $$c_{ij} = \sum_{k=1}^{r} a_{ik} \times b_{kj}$$

- The number of operations required to multiply A X B is $m \times n \times (2r-1)$

- For simplicity, usually it is analyzed in terms of square matrices of order n. So that the quantity of basic operations between scalars is

  $$2n^3 - n^2 = O(n^3)$$

Implementation of Parallel Algorithm:

Consider two square matrices A and B of size n that have to be multiplied:

1. Partition these matrices in square blocks p, where p is the number of processes available.
2. Create a matrix of size $p^{1/2} \times p^{1/2}$, so that each process can maintain a block of A matrix and a block of B matrix.
3. Each block is sent to each process, and the copied sub blocks are multiplied together and the results added to the partial results in the C sub-blocks.
4. The A sub-blocks are rolled one step to the left and the B sub-blocks are rolled one step upward.
5. Repeat steps 3 and 4 sqrt(p) times.

Cannon’s algorithm

```
for all (i=0 to s-1)        ...  "skew" A
   Left-circular-shift row i of A by i,
       so that A(i,j) is overwritten by A(i, (j+i) mod s)
end for
for all (i=0 to s-1)        ... "skew" B
   Up-circular-shift column i of B by i,
       so that B(i,j) is overwritten by B( (i+j) mod s, j)

end for
for k=0 to s-1
   for all (i=0 to s-1, j=0 to s-1)
       C(i,j) = C(i,j) + A(i,j)*B(i,j)
       Left-circular-shift each row of A by 1,
           so that A(i,j) is overwritten by A(i, (j+1) mod s)
       Up-circular-shift each column of B by 1,
           so that B(i,j) is overwritten by B( (i+1) mod s, j)
   end for
end for
```

Performance

- Skewing A. (s/2)*(alpha + (n/s)^2*beta) = sqrt(p)*alpha/2 + n^2/(2*sqrt(p))*beta
- Skewing B. The cost is the same as for skewing A.
- Shifting A (or B) left (or up) by 1. This costs alpha + n^2/p*beta.
- Local accumulation of A times B into C. This costs 2*(n/s)^3 = 2*n^3/p^(3/2).
- The total cost is therefore,

```
Time = 2*n^3/p + 3*sqrt(p)*alpha + 3*n^2/sqrt(p) * beta
```

Assumptions

- For simplicity, worked with square matrices.
- Size of the matrix must be divisible by square root of available machines (for block division).

Implementation

- **Language:** Python
- **Libraries:** Numpy and Pyro4
- Client program takes following arguments
    - Available Machines
    - Dimension of matrices
- Server program takes following arguments
        - Port number

Scope

- Generalizing to matrix of any size.
- Synchronization when there are latency issues in the network.

Resources

1. https://cse.buffalo.edu/faculty/miller/Courses/CSE633/Ortega-Fall-2012-CSE633.pdf
2. https://people.eecs.berkeley.edu/~demmel/cs267/lecture11/lecture11.html

Thanks

- Source code is at https://github.com/raghu5910/distributed_matrix_multiplication