ps5

Malvika Rajeev

10/15/2018

Question 4

Assume you have p cores with which to do the computation.

(a) Consider trying to parallelize matrix multiplication, in particular the computation XY where both X and Y are n x n. There are lots of ways we can break up the computations, but let's keep it simple and consider parallelizing over the columns of Y . Given the considerations we discussed in Unit 8, when you parallelize the matrix multiplication, why might it be better to break up Y into p blocks of m = n/p columns rather than into n individual column-wise computations?

Ideally we have no more processes than cores on a node. If there are many small tasks, then theres going to be an associated communication cost of starting and stopping the tasks, which will reduce efficiency. So every node will need to start up m times in the individual column-wise approach. The m = n/p approach would mean that each core starts and stops only once.

(b) Let's consider two ways of parallelizing the computation and count (1) the amount of memory used at any single moment in time, when all p cores are doing their calculations, including memory use in storing the result and (2) the communication cost. Count the total number of numbers that need to be passed to the workers as well as the numbers passed from the workers back to the master when returning the result. Which approach is better for minimizing memory use and which for minimizing communication?

(1) the amount of memory used in any single amount of time: Approach A: Total of p tasks, so amount of memory sued will be of storing two matrices (an nxn matrix and a nxm matrix) and then its product, so an nxm matrix.

Approach B: Total of (p^2) tasks, so amount of memory used will be, at any amount of time, of storing two matrices of size mxm and its product of size m x m.

So for use of memory, approach A is better.

(2) The communication cost - the total number of numbers that need to be passed to the workers as well as the number passed from the workers back to the master when returning the result. Approach A: The communication cost is lower because each node needs to start up only once. It gets a two matrices (X and the ith column of Y), multiplies it, and passes the resulting matrix on to the master.

Approach B: Communication cost will be higher: each core will have to communicate p matrices to the master node, and starting and stopping p times. Also compute each submatrix, information about all the previous submatrices is needed.

Question 1

The determinant of the product of matrices is the product of the determinants. Also, the determinant of an orthogonal matrix is 1 or -1.

So the product of the determinants of

$$\Gamma$$

and

$$\Gamma^T$$

is going to be 1. (The determinant of a matrix and its transpose is equal.)

So basically the right hand side boils down to the determinant of the diagonal matrix U, which is the product of the diagonal elements (which are eigen values). And

$$|A - \lambda| = 0 for each \lambda.$$

So the determinant of A is essentially the product of all the eigen values, which is the right hand side.

Question 2

The best way to approach this problem would be that for positive numbers, we divide the numerator and the denominator by so we get

$$\frac{1}{1 + e^{-z}}$$

So that when we have larger values of z, the exponential term will approach 0.

And for negative numbers, the function can be expressed as it is.

Question 3

Consider the following estimate of the variance of a set of large numbers with small variance. Explain why these two estimates agree to only 5 digits when mathematically the variance of z and the variance of x are exactly the same

```r
set.seed(1)
z <- rnorm(10, 0, 1)
x <- z + 1e12
formatC(var(z), 20, format = 'f')
```

```
## [1] "0.60931443706111987346"
```

```r
formatC(var(x), 20, format = 'f')
```

```
## [1] "0.60931216345893013386"
```

R follows a degree of precision upto 16 digits. Each term of z has 8 decimal places, and add that to 1e12 we get 20 digits. So theoretically the variance should be the same, but for R the last 4 decimal places are never considered when doing calculations, hence the disconnect. Precision is actually attained upto 4 terms.