UROP 1100 Report

Cloud Computing Algorithm

Shang Hang

1. Calculate Pi

Several ways to calculate pi: After searching , I found that there are mainly two methods for calculating pi.

1) Using the BBP(Bailey–Borwein–Plouffe formula)

Compute the nth binary digits of pi. The formula can directly calculate the value of any given digit of π without calculating the preceding digits.

pi = 3.1415926535 8979323846 2643383279……

hex = 3.243F6A88 85A308D3 13198A2E …..

= 11.00100100 00111111 01101010 …..

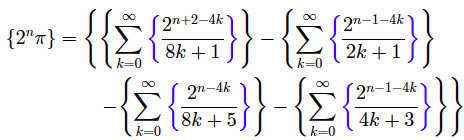
according to the formula:



so by multiply by 2^n and then we take the fraction part

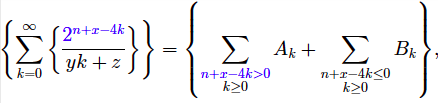
actually it is moving the n digits. So we can extract the n+1 th digits of pi directly.

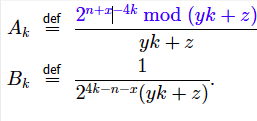
To simplify the calculation, we can drop the integer part earlier and simply calculate the fraction part.



then calculate each of the four sum.

Bacause the calculation of the four sum is separated, and within each sum we can further divide it into two sum :



one is n+x-4k>0, then we take the mod then divide to drop the integer, when n+x-4k<0, directly divide. So we got 

so these two part can also be calculated separately. The running time of this algorithm is : (n for position ,p for precision.)

Macintosh HD:Users:haozaijie:Library:Group Containers:Q79WDW8YH9.com.evernote.Evernote:Evernote:quick-note:haozaijie___Evernote:quick-note-EbOH5h:attachment--nSDvPr:screenshot.png

It is easy to distribute the work and compute in parallel.

Another algorithm is Bellard formula, which is more faster than BBP but it’s the same idea and similar step.

For Ak, we need to do the mod efficiently because 16^(n+k) might be quit large, so I think using the binary exponentiation algorithm would be better, thus in this way we can simply compare the n+k with a set of numbers indicates the powers and do I^2k = (mod(I^k)^2) mod like that and finally lower the computation to a relatively low level.

And for the Bk term ,since it is very small, cause it is 1/(16^n+k\*jk+m), so maybe after it is smaller than a presision we can omit the Bk terms. The Value of the Threshold may be discussed later. At the first guess, I think if we have a precision of 16 then 1/16^16 -> 1^-20 would be ok.

2.Monte Carle Method to calculate pi

This is and statistical simulation method to calculate pi.

1. We generate a great amout of random number within a square with width 2r.
2. Calculate the distance between the point and the origin to check whether it is <=r;
3. Count the amount of all the points and the points inside/ on the circle. Divide them then we got the value of pi;

For this method, if we got more points we got more precision and we are also able to distribute the work to different machine and run parallel, simply let them generate random numbers and report the total amount and the amount of points within the circle to the master.

I think this method has a limitation that the random number generated has problem because the computer cannot generate real random number, say if we use a cluster of same high-performance/ cost computers. The algorithm for random might be same therefore the seed might be same. So when parallel computing the result is difficult to determine whether is reliable or not.

3. continue working on the BBP method:

Map-side job : divide it into m maps and each map evaluate a set of sum: contains multiple mappers to execute. series(c, m) compute s the m part of the sum .

Reduce: compute the final sum; similar multiple reducer to calculate the intermediate value and do centralized summation. Reduce (c , sum)

//Maybe Further divide the jobs :total =jobs -> tasks -> threads

Implement in Puc:

To distribute the work, use a series() function to calculate the independent series of the equation(map) the series was assigned with the data segement s[] to store the sum of its part and a standalone array tp[] that store the exponentials to compare with in order to do the modulate.

The sum() function was used to sum the total digits(reduce) at the primary stage I only assign one reduce function. Actually in the four series, it could be further divided to calculate the two terms in one series thus concatenate two map – reduce. The sum() using s[] and watching a array ep[] which indicates whether each part of the work has been done or not, once all the map function has done it would start sum all the parts.

Expm() function is to calculate the modulate of 16^p%(ek+y), it use the binary exponentiation algorithm to mod, using a standalone array to compare with p and found the result without really doing pow(), this method could effectively reduce the work to a acceptable scale.

Hex(), change the 16 digits into hex format and output ()

Main function, run the series() and let sum watching the result and finally output the result.