CODE:

```
#include <iostream>
#include <math.h>
#include <time.h>
#include <iomanip>
using namespace std;

//function declaration
double calculusOne (int n, double xDelta);
double calculusTwo (int n, double xDelta);

int main () {



   clock_t t;
   int precision [] =  {2, 20, 200, 2000, 20000, 200000};
   double upperLimit = 1.0;
   double lowerLimit = 0.0;
   double integralInterval = upperLimit - lowerLimit;
   cout << "Calculus One" <<endl;
   for (int i = 0; i <= 5; i++) {
      double xDelta = integralInterval / precision[i];
      t = clock();
      double piValue = calculusOne(precision[i], xDelta);
      cout << "Time: "<< (((float)(clock() - t))/CLOCKS_PER_SEC) * 1000.0 <<" milliseconds"
<<endl;
      cout << setprecision(12)<< piValue <<endl;
   }
   cout << "Calculus Two" <<endl;
   for (int i = 0; i <= 5; i++) {
      double xDelta = integralInterval / precision[i];
      t = clock();
      double piValue = calculusTwo(precision[i], xDelta);
      cout << "Time: "<< (((float)(clock() - t))/CLOCKS_PER_SEC) * 1000.0 <<" milliseconds"
<<endl;
      cout << piValue <<endl;
   }
   return 0;
}

double calculusOne (int n, double xDelta) {
```

```
   double xValue = 0.0;
   double pi = 0.0;
   for (int i = 1; i <= n; i++) {
      //find height using f(x)
      double height = 4.0 / (1.0 + (xValue * xValue));
      double area = xDelta * height;
      pi += area;
      xValue += xDelta;
   }
   return pi;
}

double calculusTwo (int n, double xDelta) {
   double xValue = 0.0;
   double pi = 0.0;
   for (int i = 1; i <= n; i++) {
      //find height using f(x)
      double height = 4.0 * sqrt(1.0 - (xValue * xValue));
      double area = xDelta * height;
      pi += area;
      xValue += xDelta;
   }
   return pi;
}
```

RESULT:

Input Size	Estimated Pi Value (Calc. 1)	Estimated Pi Value (Calc. 2)
2	3.6	3.73205080757
20	3.19117598695	3.22846487975
200	3.14658848692	3.15117694484
2000	3.14209261192	3.14257950591
20000	3.14164265317	3.14169223782
200000	3.14159765359	3.14160264044

DISCUSSION:

Accuracy is directly proportional to the value of N. As the size of n is increased, the accuracy of the estimation increases as well.

Runtime is directly proportional to the value of N. As the size of n increases, the computational time increases as well.

To solve the problem in parallel, I will split the problem into groups of rectangles since each rectangle area is not dependent on each other. In the case of 20 rectangles, assuming I have 4 processors, Processor 0 can compute the area of first 5 rectangles, Processor 1 calculates the second group of rectangles and so does Processors 2 and 3. At the end of the calculations, all other processors except Processor 0 will send their result to Processor 0. Processor 0 can then add them up and print the estimated value of Pi.