Mhamed Beghdadi

760301-9252

Poor Mans Parallelism

Intruduction

The problem is solved using C#.

Two applications are created, a server application accept request over TCP network,

and a client application wich devide the work load and spread it to the servers.

The servers in this case are all the locla machines and they are listening from different ports.

The solution:

The client application

To establish communications, TCP synchronized sockets are implemented, a Class Connect that has the method to perform the send operation.

The main program

1.The system prompt for specification of the problem, the user enter the values of C,sizes x, y, max number of iterations max_n, number of servers, the addresses and port numbers,

this is stored in a string.

2. Extract values from the string using the split method and store the values in a

string array data_str;

3. Convert string obtained to integers and doubles.

4. Devide the work.

This is done in th following steps:

- determine the optimal division of servers such as each server get a workload, such that we get a aX b

division (a horizontally and b verticaly), that is divs = a x b.

In the case divs is a prime number, the total number of servers divs is a prime number the we take a = divs and b=1.

In the case it is not, a is determined to be the first prime number that divide the total number divs.

- determine the work load, each server get the amount of data specified as xloc and yloc. Then xloc = x/a, ylo c = y/b.

- set the byte array where the global picture is stored as Entire_pic.

- set the byte array where local picture is stored as part.

-Prepare connections.

The client should send requests to servers synchronically, that is no send-receive operation should be first terminated to connect for the other server.

To implement this, thread are used. Threads take delegates as objects, these delegates perform the sed operation by calling the method SendWork of a Class Connect.

To begin the operation, the thread are started byt the methos Start():

The class Connect is a class which has the constructor getwork, the constructor take the values of min and max of C, local part xloc and yloc, max_n, host adress of the server and port , and i, j wich the determine the where in the global picture the local part is situated.

The class Connect has a private member part, where the result from the server is stored, when the receive operation is terminated.

For example for i=0,j=0 the part is located in the upper left corner of the global picture, for i=0,j=1 the part is theupper second part horizontally.

-To send the work loads, we iterate on i and j.

- different servers take different times to perform the calculations. The class Connect has a private boolean member status which is set to true when the data is received.

- when all servers have finiched the send back operation, the message "all servers responded." is displayed.

- Iterate on all obtained parts and store the data in he global picture variable entrie-pic.

- Convert the entire_pic variable to a 2-dimensinal array of integers.

-Store the result in a PGM file.

SendWork() methos in the class Connect:

This method sends the information using a socket.

- establish the remote endpoint for the socket.

The IP address is obtained by parsing the host, using the method Parse.

The remode endpoint is obtained by the Ip address and the port number.

Create the TCP/IP socket sender.

Connect the socket to the reomote endpoint by the operation send.Connect(remoteEP).

A string that contains all information to send is constructed by converting the the values to string values and then by the method Concat join them alltogether, the string work is obtained.

Convert the string to bytes and store it as a variable msg.

Send the data to the socket by the operation sender.send(msg) and the receive the response from the server by the operation send.receive(bytes), where bytes is a byte buffer.

Close the socket and set the boolean variable status to true.

Catch error by throwing exceptions, and display the errors when catched.

The server application

The server application uses sockets to listen for incoming communications from the client.

The main program

Preparation of servers for listening, this is done by the following steps:

1. Create instances of the Class Connect, that is connections, the class has a constructor serverspec, which take the hos address and the port number as arguments.

2. Create threads, as the listening operation is performed by each server independantly, threads are used, they take delegates as ThreadStart and these delegates take a method of the class which is listen() to begin listening for incoming connections.

3. Threads are started, the method listen() are called.

The method listen() of the class Connect:

The method obtain the IP address by parsing the host, then establish a local endpoint for the socket, then create the TCP/IP socket listener.

Bind the socket to the local endpoint and begin listening, this is done by the operations:

```
listener.Bind(localEndPoint);
listener.Listen(10);
```

Extract the first pending connection request from the connection request queue of the listening socket, and then creates and return a new socket handler:

socket handler = listener(accept);

The connections is processed, bytes are received by the operation

int bytesRec = handler.Receive(bytes);

Convert the bytes received to a string data,

Get the substring data_new which contains the values of C and xloc,yloc, max_n and the position i,j.

Convert the string values extracted as integers and doubles.

Iterate by values of x and y that varies of the partial region horizontally and vertically. Iterate by valus of C, from minC_re to maxC_re and from minC_im to maxC_im.

For each pixel Z0 with coordinates (x,y) in in the partial region we have the intial values:

```
// initial values
double tx = x + cx;
double ty = y + cy;
```

That is Z1 with coordinates (tx,ty) Z1 = Z0 + C .

We calculate the absolute value as the following:

```
double znewx = tx * tx - ty * ty + tx;     The power
double znewy = ty + 2 * tx * ty;
double absZ = Math.Sqrt(znewx * znewx - znewy*znewy);  The square root.
```

That is to check if the absolute value is greater then the abslute value of C, as the following:

```
if (absZ > Math.Sqrt(cx * cx - cy * cy))
```

If this is true, then stop the iteration and n is the number of iterations performed modulo 256, convert then to bytes and store in the partial array part_pic as the follwing:

```
byte pxl = (byte)(n % 256);
part_pic[(x - i * xloc) + y * (yloc - j * yloc)] = pxl;
break;
```

Otherwise, continue the iteration, and the new initial values are

```
else n++;
tx = znewx;
ty = znewy;
```

That is Z2 = Z1 + C .

The iteration continues in this manner until n reaches the maximum number of iterations allowed, max_n.

we set this as the follwing:

```
while (n < max_n);
```

When the iteration is performed over all points in the partial region, we obtain a byte array that contains the number of iterations performed foreach pixel.

The data part_pic is sent back to client, and socked is closed. Catch exceptions and dosplay errors if any exists.