학기말 Project : 분산 컴퓨팅

과목 : PC 기반제어프로그래밍

교수님 : 김성환 교수님

제출일자 : 6/26/금

학번 : 16100166

성명 : 윤인재

목차

1. 통신 프로토콜 설계

크게 두 가지로 분류하자면

1) Client - 소수 계산을 시키는 명령 및 정보를 받는 통신

2) Server – 명령에 대한 행동 및 정보를 보내는 통신

Client에서 Server로 또는 반대로 명령을 줄 때 이벤트 핸들링을 함수로 이용하여 설계하면

Ex) **Client** **Server**

TClient clientCalculate → TServer serverCalculate(**이벤트 함수**)

TClient clientRcv(**이벤트 함수**) ← TServer serverSendResult

```
//Client
private TClient clientCalculate1; //서버 1에 계산 명령 보내기
private TClient clientRcv1;//서버 1로부터 소수 값 받기
private TClient clientCalculate2; //서버 2에 계산 명령 보내기
private TClient clientRcv2;//서버 2로부터 소수 값 받기
...
if (clientCalculate1 == null) clientCalculate1 = new TClient();
            clientCalculate1.ClientBeginConnect(serverIP, 5001, clientIP);

if (clientRcv1 == null) clientRcv1 = new TClient(Svr1_PimeNumberDataArrived);
            clientRcv1.ClientBeginConnect(serverIP, 5002, clientIP);

if (clientCalculate2 == null) clientCalculate2 = new TClient();
            clientCalculate2.ClientBeginConnect(serverIP, 5003, clientIP);

if (clientRcv2 == null) clientRcv2 = new TClient(Svr2_PimeNumberDataArrived);
            clientRcv2.ClientBeginConnect(serverIP, 5004, clientIP);
```

```
//Server 1
private TServer serverCalculate1; //계산 명령 수신용
private TServer serverSendResult1;// 결과 송신용
…
if (serverCalculate1 == null) serverCalculate1 = new TServer(GetRange);
            serverCalculate1.ServerStartListen(myIP, 5001);
if (serverSendResult1 == null) serverSendResult1 = new TServer();
            serverSendResult1.ServerStartListen(myIP, 5002);
```

```
//Server 2
private TServer serverCalculate2; //계산 명령 수신용
private TServer serverSendResult2;// 결과 송신용
…
if (serverCalculate2 == null) serverCalculate2 = new TServer(GetRange);
            serverCalculate2.ServerStartListen(myIP, 5003);
if (serverSendResult2 == null) serverSendResult2 = new TServer();
            serverSendResult2.ServerStartListen(myIP, 5004);
```

총 4개의 프로토콜을 설계했습니다.

1) 세부 설계과정

① Client 에서 명령을 내리는 버튼을 누를 때 발생되는 주요 코드

```
//input 과 step() server_1,2로 송신
   int input = Convert.ToInt32(InPut.Text);
    int step = xmax/ PicArea.Width; //xmax = 30,000,000
    txtstep.Text = Convert.ToString(step);
    string st1 = TSocket.sSTX() + Convert.ToString(input)
                                + "," + Convert.ToString(step) + TSocket.sETX();
    clientCalculate1.ClientSend(st1);
     clientCalculate2.ClientSend(st1);
```

범위 기준값(Input)과 해상도에 맞는 Step을 선언하여 string형으로 서버1과 서버2에 전송했습니다.

② 명령을 전달받은 Server 측의 이벤트 함수 코드(Server 1 과 Server 2 는 동일코드 사용)

```
private void btnListen_Click_1(object sender, EventArgs e)
{
if (serverCalculate1 == null) serverCalculate1 = new TServer(GetRange);
            serverCalculate1.ServerStartListen(myIP, 5001);
```

```
//클라이언트의 Prime Input 수신
        private void GetRange()
        {
            while (true)
            {
                //범위 버퍼에 메시지 저장
                string rbuffrange = serverCalculate1.GetRcvMsg();
                int idx1 = rbuffrange.IndexOf(TSocket.sSTX());
                if (idx1 < 0) break;
                int idx2 = rbuffrange.IndexOf(TSocket.sETX(), idx1);

                if (idx1 >= 0 && idx2 > idx1)
                {
                    string range = rbuffrange.Substring(idx1 + 1, idx2 - idx1 - 1);
                    char[] sep = new char[] { ',' };
                    string[] xy = range.Split(sep);
                    Primerange = Convert.ToInt32(xy[0]);
                    step = Convert.ToInt32(xy[1]);

                    rbuffrange = rbuffrange.Substring(idx2 + 1);
                    txtNMax.Text = Convert.ToString(Primerange);
                }
                else
                    break;
            }
            //계산 실행
            Calculate(Primerange, step);
        }
```

①계산할 소수의 범위(Primerange)와 Step정보를 저장 후 Calculate 함수로 처리했습니다.

②Calculate함수 알고리즘 분석은 주요코드 설명에서 다루겠습니다.

③ Server에서 Client로 보내는 송신코드(Calculate 함수 내부)

```
private void Calculate(int Primerange, int step)
      {
           …
           DateTime stime = DateTime.Now; //측정 시작
           for (int i = 2; i <= Primerange; i += step)
           {
               PrimeCSharp.FindNumberOfPrimeNumber(NumStart, i, out nprime);
               …
               …
               // x값과 pi(x)값 보내기
               string result = TSocket.sSTX() + Convert.ToString(i) +
                       "," + Convert.ToString(SaveNum) + TSocket.sETX();
               serverSendResult1.ServerSend(result);
           }
           double dtime = Util.TimeInSeconds(stime); //시간 측정
           //시간과 종료를 알리는 EXIT 값을 보낸다.
           lblTime.Text = string.Format("{0:0.00}", dtime)+" sec";
           string Call_time = TSocket.sSTX() + lblTime.Text +
                           "," + "EXIT" + TSocket.sETX();
           serverSendResult1.ServerSend(Call_time);
       }
```

서버에서 클라이언트 측으로의 송신은 소수를 계산할 때마다 x값과 pi(x)값을 실시간으로 보내고 계산이 끝나면 걸린 시간과 끝을 알리는 "EXIT" 문자열을 송신합니다.

④ 서버로부터 문자열을 수신한 Client부 이벤트 함수 코드

```
if (clientRcv1 == null) clientRcv1 = new TClient(Svr1_PimeNumberDataArrived);
            clientRcv1.ClientBeginConnect(serverIP, 5002, clientIP);
```

```
// server1의 x값 , pi(x), 시간 수신
      private void Svr1_PimeNumberDataArrived()
      {
          …
          //Server1의 x값, pi(x)값 수신
          while (true)
          {
              rbuffSvr1 = clientRcv1.GetRcvMsg();
              …
                  //"EXIT"를 입력받으면 Server1의 계산시간 수신 및 그래프 그리기
                  if (xy[1] == "EXIT")
                  {
                      …
                  }
                  // 2X2 배열에 저장
                  Index1[svr1_i, 0] = Convert.ToInt32(xy[0]);
                  Index1[svr1_i, 1] = Convert.ToInt32(xy[1]);
                  …
```

```
                    svr1_i++;
                    rbuffSvr1 = rbuffSvr1.Substring(idx2 + 1);
                }
                else
                    break;
            }
        }
```

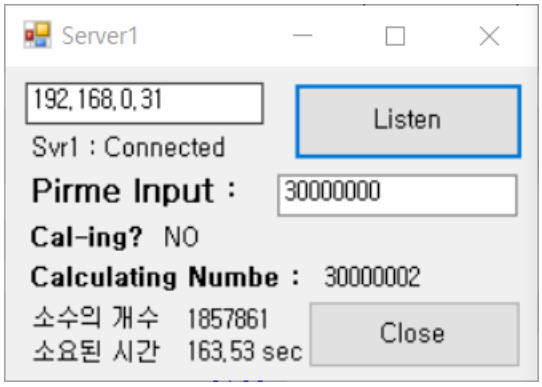
①서버의 for구문으로부터 x값과 pi(x)값을 수신하면서 2X2배열에 저장시킵니다. 마지막에 계산시간과 "EXIT" 문자를 입력 받으면 배열 저장값으로 Client에 시간과, 그래프를 표기합니다.
②정리하면 Client 명령→ Server 응답 및 계산→ Server 송신 → Client 수신 과정입니다.

2 번 설명에 앞서서 프로젝트 한글파일에서

- $\pi(2)=1$ 이고 $\pi(30,000,000)=1,857,859$ 이다. 일반 노트북에서 C#으로 이를 계산하는데 약 50초 가량이 소요된다.

소수 판별을 sqrt()+1 으로 시간을 충분히 줄였음에도, 제 노트북은 30,000,000 을 일반적으로 계산하는데 163.53 초가 걸렸습니다. 노트북을 산지 8 년이 다되어서 성능이 좋지 못한 점 감안해주시면 감사하겠습니다.

- **단일 서버로 구했을 때**

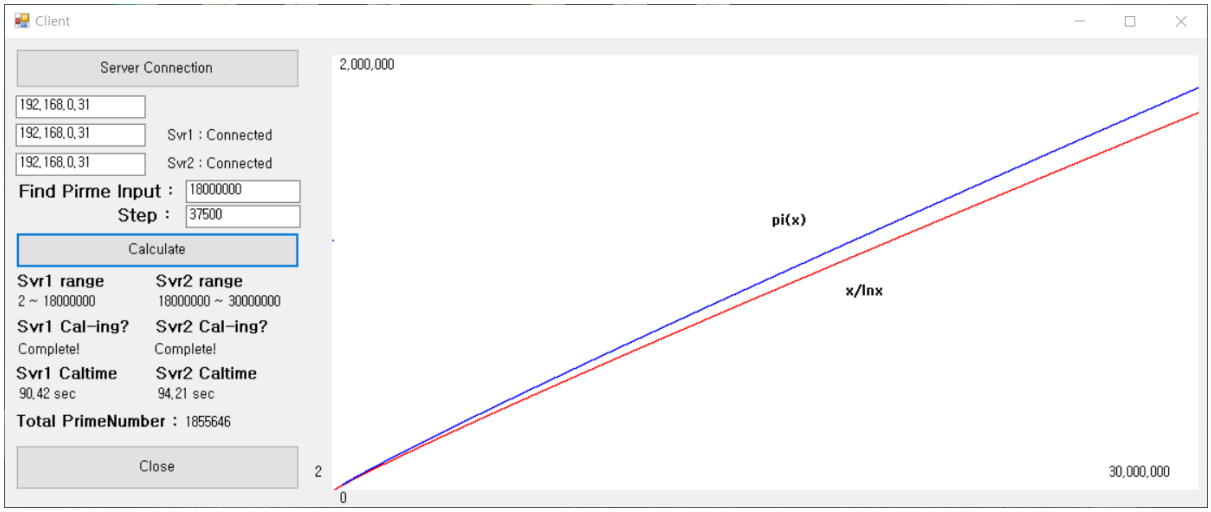


2. 그래프 및 범위에 따른 시간 비교

1) 15,000,000 을 기준으로 입력했을 때 → Svr1 = 69.28 초, Svr2 = 110.71 초

2) 17,000,000 을 기준으로 입력했을때 → Svr1 = 87.60 초, Svr2 = 104.45 초

3) 18,000,000 을 기준으로 입력했을때 → Svr1 = 90.42 초, Svr2 = 94.21 초



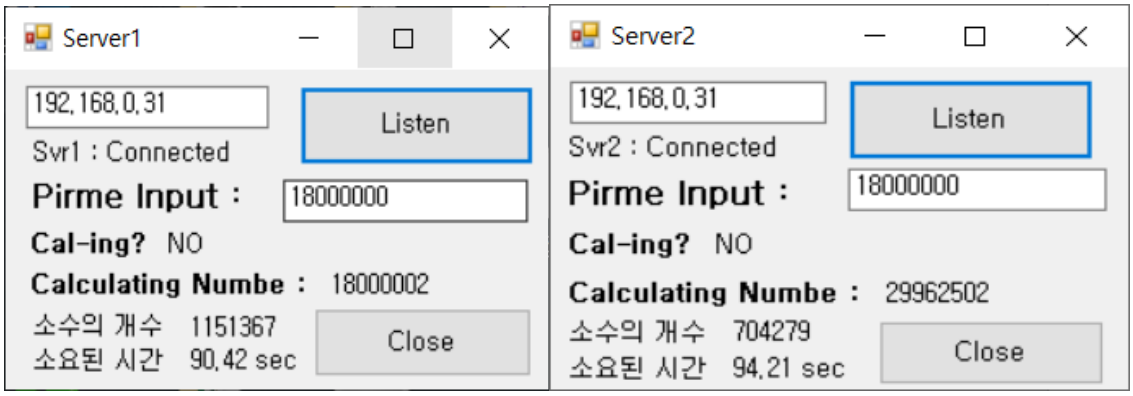
1) Client, Server1, Server2의 IP주소를 받고 Server Connection으로 통신을 연결했습니다.

2) PictureBox의 크기는 800*600으로 프로젝트 한글파일의 소수값과 맞는지 비교하기위해 선정했습니다.

3) 18,000,000을 기준으로 나누었을 때 Svr1 = 90.42초, Svr2 = 94.21초로 가장 비슷한 값이 나왔습니다.

4) 30,000,000을 일반적으로 계산했을시 163.53초가 나왔는데 분산을 통해 약 70초가량 단축했습니다.

3. Server 1 , Server 2 결과 창

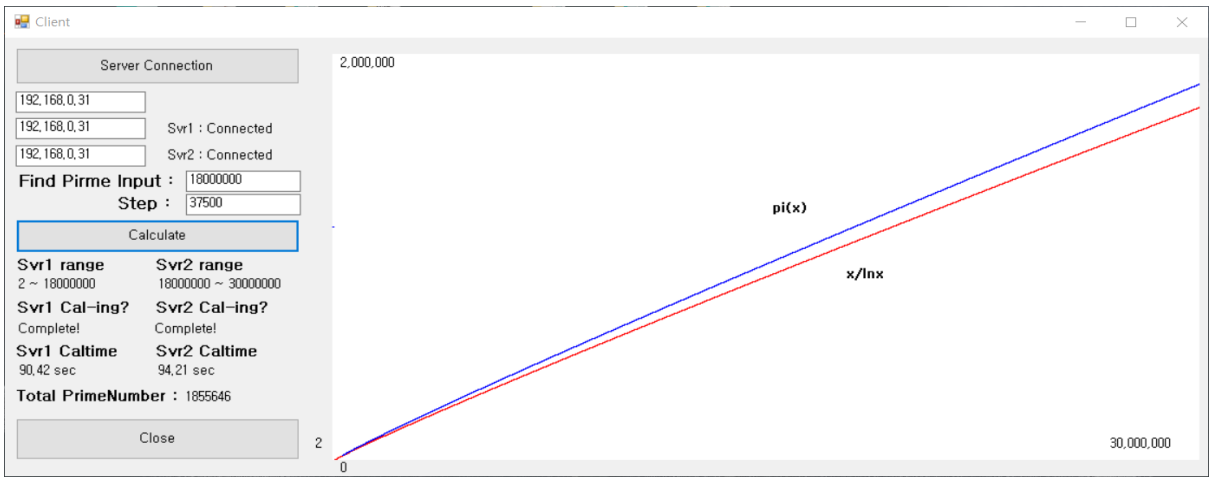


1) 컴파일시 IP는 바로 뜨며 Listen 버튼으로 통신을 시도합니다.

2) 입력을 받고 계산할때 실시간으로 x값, pi(x)값이 나오도록 했습니다.

3) Server 1의 18,000,000 인풋에 따른 소수의 개수는 1,151,367이 맞습니다. 하지만 Server 2의 소수의 개수는 704,279이 아닙니다. 그 이유는 18,000,002부터 interval 간격으로 37,500씩 증가할 때마다 그 사이의 소수의 개수를 측정하고 저장하기 때문입니다. 마지막에 그래프를 나타낼 때 Server2에서 계산한 소수값에 Server1의 마지막 소수의 개수를 더하여 그래프를 나타냈습니다.

4) Client 결과창



3. 소스 코드의 주요 부위에 대한 설명

1) Form_Load시 IP 표시

```
private void Form1_Load(object sender, EventArgs e)
        {
            txtServerIP.Text = TSocket.HostAddresses()[1].ToString();//XP는 [0]
        }
```

2) xpixel화 ypixel화 함수

```
int xmin = 0, ymin = 2;
int xmax = 30000000, ymax = 2000000;
```

```
private float xpixel(double xw)
        {
           return (float)(PicArea.ClientSize.Width * (xw - xmin) / (xmax - xmin));
        }
        private float ypixel(double yw)
        {
           return (float)(PicArea.ClientSize.Height * (1 - (yw - ymin) / (ymax -ymin)));
        }
```

4주차 과제 LeastSquare에서 배운 함수를 이용했습니다.

3) 명령을 내리는 버튼 클릭 이벤트 발생시 $\frac{x}{lnx}$ 그래프 표시

```
Graphics grp = PicArea.CreateGraphics();
            lblxlnx.Text = "x/lnx";
            for (int k = 0; k <= PicArea.Width; k++)
            {
                double pic_step = xmax / PicArea.Width;
                grp.DrawEllipse(new Pen(Color.Red), xpixel(k * pic_step),
                                ypixel(k * pic_step / Math.Log(k * pic_step)), 1, 1);
            }
```

PictureBox에 대한 Step을 만든 후 pixel 함수로 x/lnx 그래프를 그렸습니다.

4) timConStatus_Tick으로 통신이 연결될 시 서버 IP 동기화

```
private void timmConnStatus_Tick(object sender, EventArgs e)
        {
            //Server_1
            if (clientCalculate1 == null) { lblSvr1.Text = "Cal1 : " + "NULL"; }
            else
            {
                csConnStatus conn = clientCalculate1.ClientStatus();
                lblSvr1.Text = "Svr1 : " + conn.ToString();
                //Svr1 IP표시
                IPAdd1.Text = txtServerIP.Text;

            }
```

```
            //Server_2
            if (clientCalculate2 == null) { lblSvr2.Text = "Cal2 : " + "NULL"; }
            else
            {
                csConnStatus conn = clientCalculate2.ClientStatus();
                lblSvr2.Text = "Svr2 : " + conn.ToString();
                //Svr2 IP표시
                IPAdd2.Text = txtServerIP.Text;
            }
        }
```

이 부분은 솔직히 Server의 IP를 맞게 따오는지 잘 모르겠습니다. 만일 다른 IP주소에서 연결할 경우 Client에서 그에 맞는 IP주소를 입력해주어야 연결된다고 생각합니다.

5) Server의 Calculate 함수 부 (PrimeCSharp 수정 – 인수 2개 → 3개)

```
//전역변수
int Primerange; //Input값
int step; //어디서든 쓸 수 있도록 전역변수로 선언
int nprime;// 소수의 개수 pi(x)
```

```
//Server1_Calculate
        private void Calculate(int Primerange, int step)
        {
             …
            //계산시작
            int NumStart = 2;
            int SaveNum = 0;
            for (int i = 2; i <= Primerange + 2 ; i += step)//18,000,002 까지 계산
            {
                PrimeCSharp.FindNumberOfPrimeNumber(NumStart, i, out nprime);
                NumStart = i;
                if (i == 2+step) SaveNum = 0; //두번 째를 0으로 초기화한다.
                SaveNum += nprime; //직전까지의 소수값을 SaveNum에 저장
                …
}
```

① 1.Server 1에서 18,000,000을 입력으로 받았을 때 Step으로 37,500 씩 증가하면 17,962,502에서 끝나게 되는데 이러면 공백이 생기기 때문에 입력인수 Primerange에 +2 를 하여 18,000,002까지 계산되도록 했습니다.

② PrimeCharp클래스를 사용할 때 인수를 기존에 2개에서 3개로 고쳤습니다. NumStart는 한 루프 이전의 x값 입니다.

③ if (i == 2+step) SaveNum = 0;의 의미는 Server1 에서 소수 값을 띄웠을 때 2번 째부터 +1씩 되어 출력되는 것을 보고 2번째 스텝부터 SaveNum을 0으로 초기화 해주었습니다.

④ 루프를 돌때마다 pi(x)값을 저장하는 SaveNum을 추가했습니다.

```
//Server2_Calculate
       private void Calculate(int Primerange, int step)
        {
            …
            //계산시작
```

```
            int NumStart = Primerange +2; //18000002부터 계산
            int SaveNum = 0;

            for (int i = NumStart; i <= xmax; i += step)
            {
                ...
                ...
        serverSendResult2.ServerSend(Cal2_time);
        }
```

NumStart부터 계산하도록 값을 지정했습니다.

```
// PrimeCSharp Class
private static bool isPrimeNumber(int num)
        {
            bool isprime = true;
            for (int i = 2; i < (int)Math.Sqrt(num) + 1; i++)
            {
               ...
            }
            return isprime;
        }
        public static void FindNumberOfPrimeNumber(int NumStart,int nMax, out int
nprime)
        {
            nprime = 0;
            for ( int i= NumStart ; i <= nMax; i++)
            {
              ...
            }
```

좀 더 효율적인 소수 계산 방법을 찾아본 결과 자신을 루트하고 +1 만큼 한 값까지 소수에 맞는게 없으면 그 값은 소수임을 판별하도록 수정했습니다.

6) Client에서 Server1과 Serve2의 정보 저장과 그래프 표현

```
//Svr1_PrimeNumberDataArrived - server1의 x값 , pi(x), 시간 수신
//"EXIT"를 입력받으면 Server1의 계산시간 수신 및 그래프 그리기
if (xy[1] == "EXIT")
{
for (int k = 0; k < PicArea.Width; k++){
 Graphics grp = PicArea.CreateGraphics();
 grp.DrawEllipse(new Pen(Color.Blue), xpixel(Index1[k, 0]),
                                      ypixel(Index1[k, 1]), 1, 1);
  }
}
//배열에 저장
Index1[svr1_i, 0] = Convert.ToInt32(xy[0]);
Index1[svr1_i, 1] = Convert.ToInt32(xy[1]);

//Sever1의 마지막 x의 소수개수
Svr1_SaveNum = Index1[svr1_i, 1];

svr1_i++;
```

```
 }
```

① 이벤트 통신으로 2 X 2 배열에 저장 후 for 구문을 이용해 그래프를 띄웠습니다.

② Svr1_SaveNum은 Server2의 그래프를 그리기 위한 pi(x)값 입니다.

```
//Svr2PrimeNumberDataArrived - server1의 x값 , pi(x), 시간 수신
//"EXIT"를 입력받으면 Server2의 계산시간 수신 및 그래프 그리기
if (xy[1] == "EXIT")
{
lblSvr2_caltime.Text = xy[0]; //시간 표시
lblSvr2_Cal.Text = Convert.ToString("Complete!");//계산 완료 표시
TotalNum.Text = Convert.ToString(Svr1_SaveNum + Svr2_SaveNum);//최종 pi(x)값
for (int k = 0; k < PicArea.Width; k++)
{
   Graphics grp = PicArea.CreateGraphics(); // 객체 선언
   grp.DrawEllipse(new Pen(Color.Blue), xpixel(Index2[k, 0]),
                                        ypixel(Index2[k, 1] + Svr1_SaveNum), 1, 1);
}
                        for (int j = 0; j < PicArea.Width; j++)
Console.WriteLine("{0}:{1}", Index1[j,0], Index1[j,1]);
                        for (int j = 0; j < PicArea.Width; j++)
Console.WriteLine("{0}:{1}", Index2[j,0], Index2[j,1]+Svr1_SaveNum);
                    }
              …
        }
```

① PrimeRange부터 30000000까지 Step 간격으로 소수의 개수를 판별 후, Server 1에서의 마지막 pi(x)인 Svr1_SaveNum을 더해주어 매끄럽게 그려지도록 했습니다.

② 출력창에 Index2 배열을 출력할 때도 Svr1_SaveNum을 더했습니다.