Paradigmas de Computação Paralela

1º Trabalho – OpenMP : paradigma de memória partilhada

João Luís Sobral

13-Outubro-2015

1º Trabalho – OpenMP

- Objetivo:
  - Avaliar a aprendizagem do paradigma de programação baseado em memória partilhada (OpenMP)

- Parâmetros da avaliação

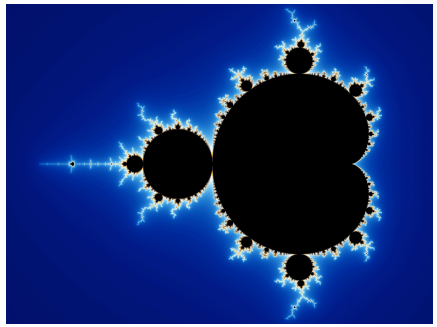
	Peso
Qualidade da implementação sequencial	10%
Desenho e implementação da versão paralela (+otimização)	30%
Qualidade (e quantidade) da experimentação (dados de entrada, métricas, medições)	20%
Análise dos resultados (justificação para valores obtidos)	20%
Relatório	20%

1º Trabalho – OpenMP

- Grupos:
  - 2 elementos (ou 1)
- Entrega:
  - 01-Nov-2015 (até as 24H)
  - Relatório (máx 4 páginas) + código
  - Email (jls@di.uminho.pt)
- Escolha dos trabalhos
  - Livre entre os vários disponíveis
  - Sugeito a arbitragem!!!!

1º Trabalho – OpenMP

- 1ª Opção: Cálculo do Fractal Mandelbrot



- Pseudo-código:

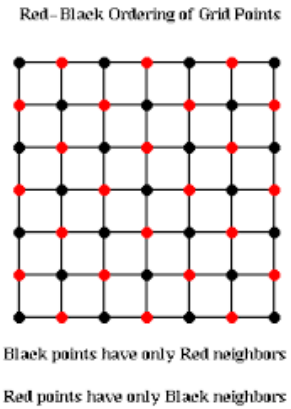
```
public void mandelBrot() {

  for (int y = y0 ; y<y0+height ; y++)  {

    float ci = c0i + ((float)y)*scale;

    for (int x = x0 ; x<x0+width ; x++) {

       float cr = c0r + ((float)x)*scale;

       pic[y-y0][x-x0] = CompPoint (cr,ci);

    }
  }

}
```

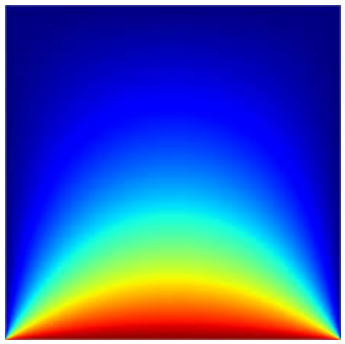
```
private int CompPoint (float cr,float ci) {

  float zr = 0.0f;
  float zi = 0.0f;
  int count = 0;

  while (count < nit  &&  zr*zr + zi*zi < 4.0f ) {
    zr = zr*zr - zi*zi + cr;
    zi = 2.0f * zr * zi + ci;
    count++;
  }
  return (count);
}
```

c0r = 0.03f;//-2.05f;
c0i = -0.5f;//-2.05f;
side= 1.025f;//4.1f;
scale = side / img_size;

1º Trabalho – OpenMP

- 2ª Opção: SOR-RB
  - Red-Black Successive Over-Relaxation (SOR) is a method of solving partial differential equations.





- Pseudo-código:

```
doIterations() {

for (int p=1; p<(2*num_iterations+1); p++) {
            oneIteration();
}
```

```
for (int i=1; i<N-1; i++) {
            for (int j=1; j<N-1; j++) {
                        Gtotal += G[i][j];
            }
}
```

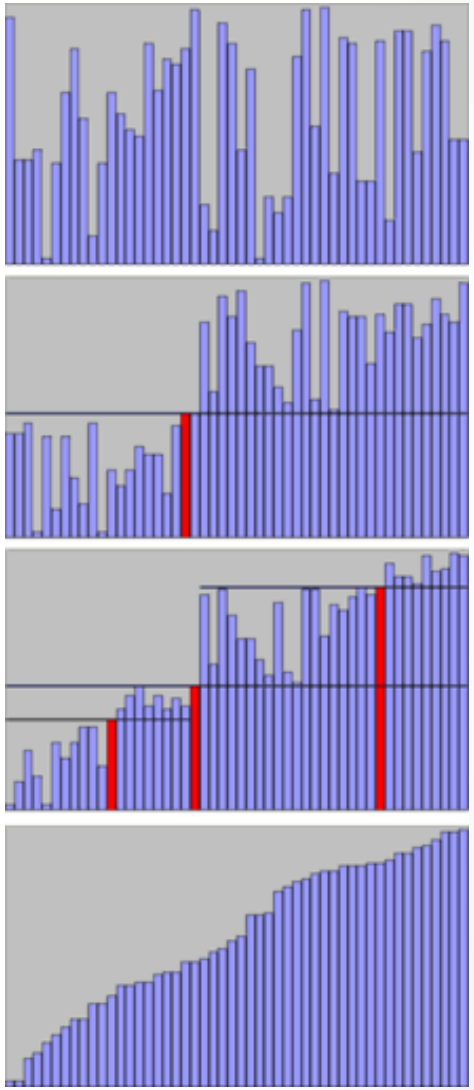
```
oneIteration() {

  for(int i=1; i<N-1; i++) {

    start = 1+i%2;

    for(int j=start; j<N; j+=2)

      G[i][j]=(omega/4.0)*(
                  G[i-1][j]+
                  G[i+1][j]+
                  G[i][j-1]+
                  G[i][j+1])
                +(1-omega)*Gi[j];
}
```

1º Trabalho – OpenMP

- 3ª Opção: Quick-sort

- Pseudo-código:

```
private void quicksort(int lo, int hi)
 {
   int i=lo,j=hi,h;
   int x=array[(lo+hi)/2];

   //partition
   do
   {
     while(array[i]<x) i++;
     while(array[j]>x) j--;
     if(i<=j)
     {
       h=array[i]; array[i]=array[j]; array[j]=h;
       i++; j--;
     }
   } while(i<=j);

   //recursion
   if(lo<j) quicksort(lo,j);
   if(i<hi) quicksort(i,hi);
 }
```



1º Trabalho – OpenMP

- 4ª Opção: Multiplicação
  Vector-matriz esparsa: y = Ax
  (formato COO)

$$\begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 0 \\ 6 & 0 & 0 & 7 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 5 \\ 1 \\ 8 \end{bmatrix} = \begin{bmatrix} 4 \\ 47 \\ 5 \\ 68 \end{bmatrix}$$

- Pseudo-código:

```
for (int i=0; i<nz; i++) {
            y[ row[i] ] += x[ col[i] ] * val[i];
}
```

Sparse Format

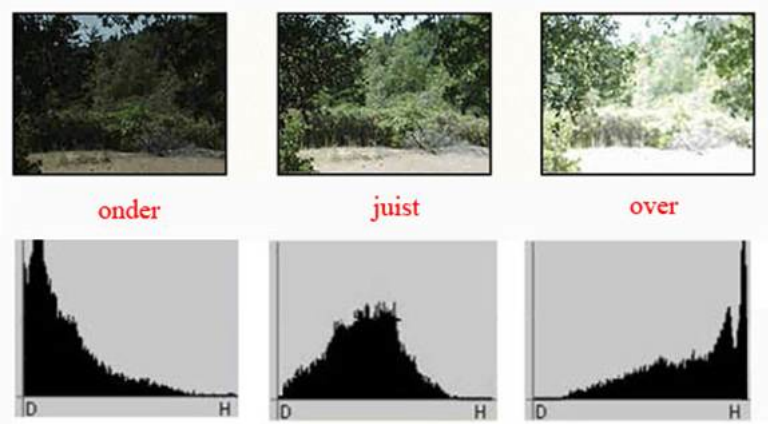
Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	...
0	0	0	0	0	...
0	0	0	0	0	...
0	0	7	0	0	...
3	0	0	0	0	...
0	0	0	0	1	⋱

Row	Column	Value
3	3	7
4	1	3
5	5	1
⋮	⋮	⋮

Coordinate List (COO) Format

1º Trabalho – OpenMP

- 5ª Opção: Equalização do Histograma de uma imagem



- Pseudo-código:

```
// contruir histograma
for (int i=0; i<size; i++) {
            histogram[ img[i] ] ++;
}

//Calcular histograma acumulado + normalizar
acc[w] = sum(histogram[level<w])

// transformar a imagem
novaimg[i] = acc[ img[i] ];
```