Използване на OpenMP. Част 3.

Курс „Паралелно програмиране“

ИНСТИТУТ *за* СЪВРЕМЕННИ
ФИЗИЧЕСКИ ИЗСЛЕДВАНИЯ

Стоян Мишев

План

Locks

Shared, private, firstprivate, lastprivate

Примери

Изпълнява се нишката, която държи lock. Останалите изчакват докато се освободи.

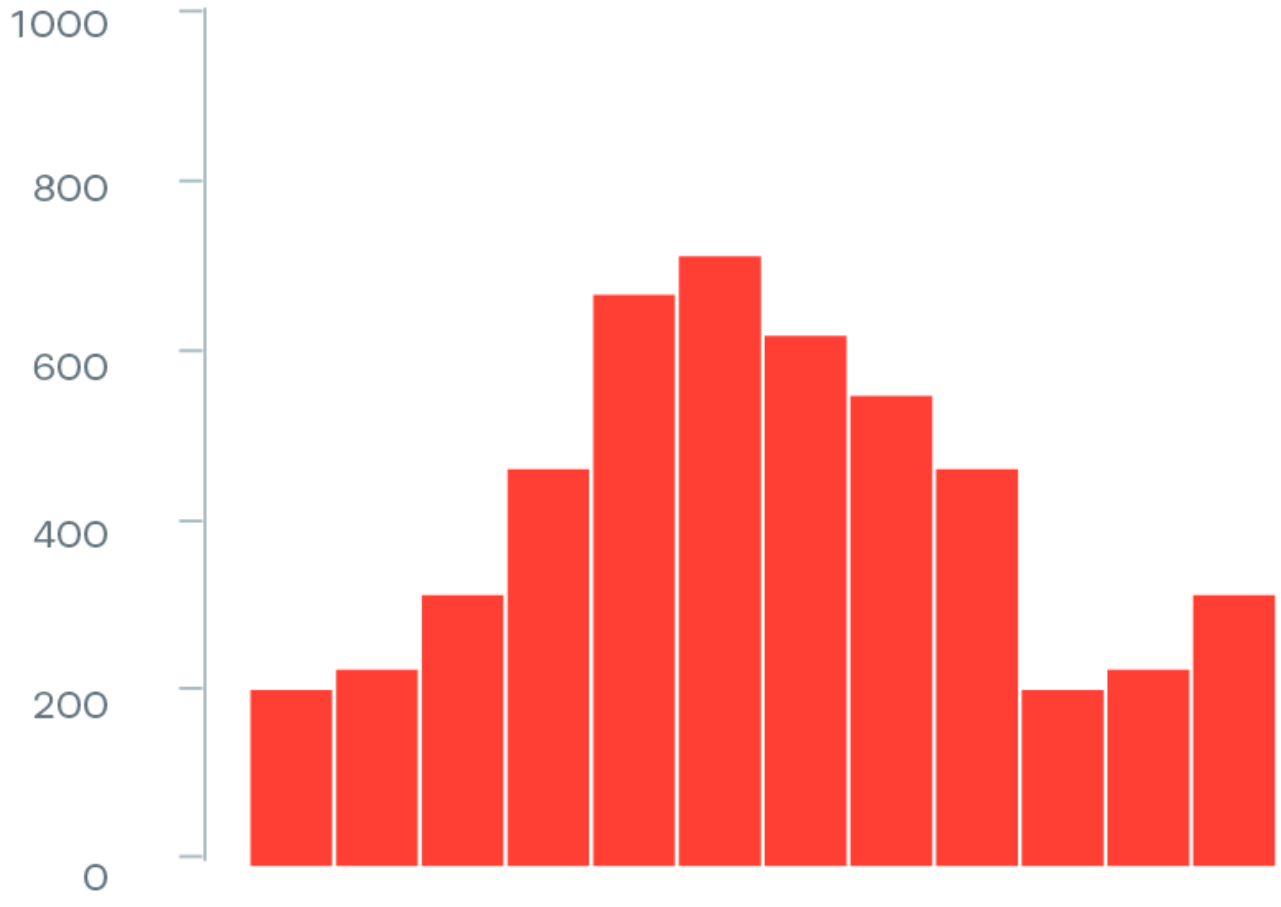
`omp_init_lock()`

`omp_set_lock()`

`omp_unset_lock()`

`omp_destroy_lock()`

Хистограма



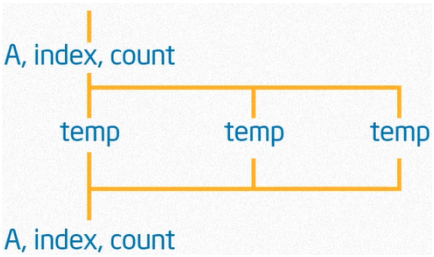
```
#pragma omp parallel for
for(i=0;iNBUCKETS; i++){
   omp_init_lock(&hist_locks[i]);    hist[i] = 0:
}
#pragma omp parallel for
for(i=0;iNVALS; i++){
   ival = (int) sample(arr[i]);
   omp_set_lock(&hist_locks[ival]);
      hist[ival]++;
   omp_unset_lock(&hist_locks[ival]);
 }

for(i=0;i<NBUCKETS; i++)
 omp_destroy_lock(&hist_locks[i]);
```

```
omp_set_num_threads()

omp_get_num_threads()

omp_get_thread_num()

omp_get_max_threads()

omp_in_parallel()

omp_set_dynamic()

omp_get_dynamic()

omp_num_procs()
```

```
#include <omp.h>
void main()
{  int num_threads;
    omp_set_dynamic( 0 );
    omp_set_num_threads( omp_num_procs() );
#pragma omp parallel
  {   int id=omp_get_thread_num();
#pragma omp single
        num_threads = omp_get_num_threads();
      do_lots_of_stuff(id);
  }
}
```

```
double A[10];
int main() {
    int index[10];
    #pragma omp parallel
        work(index);
    printf("%d\n", index[0]);
}
```

```
extern double A[10];
void work(int *index) {
  double temp[10];
  static int count;
  ...
}
```



```
shared

private

firstprivate

lastprivate

default(private|shared|none)
```

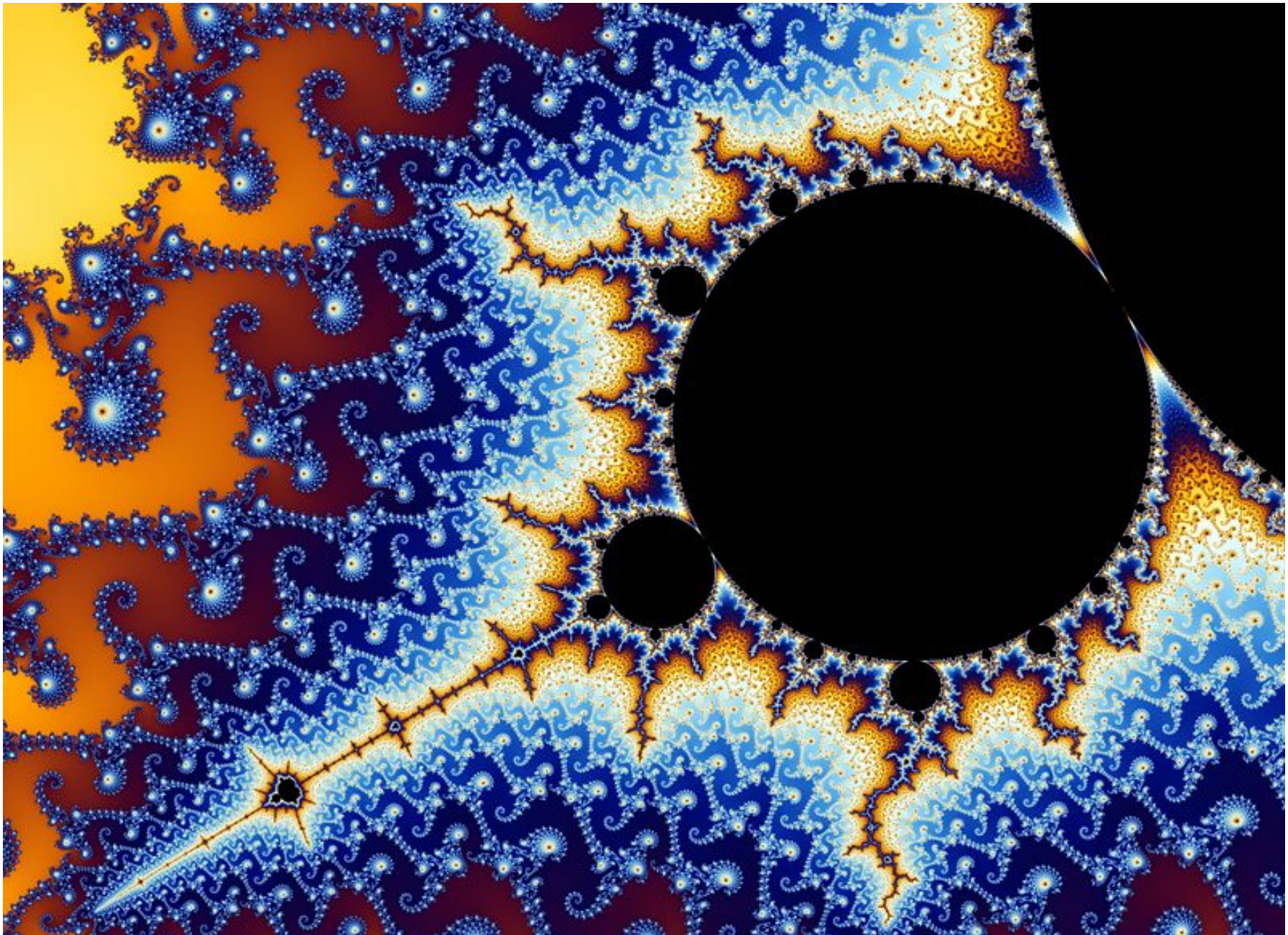
```
void wrong() {
    int tmp = 0;
#pragma omp parallel for private(tmp)
    for (int j = 0; j < 1000; ++j)
            tmp += j;
    printf("%d\n", tmp);
}
```

```
void wrong() {
    int tmp = 0;
#pragma omp parallel for private(tmp)
    for (int j = 0; j < 1000; ++j)
            tmp += j;
    printf("%d\n", tmp);
}
```

```
incr = 0;
#pragma imp parallel for firstprivate(incr)
for (i = 0; i <= MAX; i++) {
        if ((i%2)==0) incr++;
        A[i] = incr;
}
```

```
void sq2(int n, double *lastterm)
{
   double x; int i;
   #pragma omp parallel for lastprivate(x)
   for (i = 0; i < n; i++){
      x = a[i]*a[i] + b[i]*b[i];
      b[i] = sqrt(x);
   }
   *lastterm = x;
}
```

Площ на фрактала на Mandelbrot 1.



Площ на фрактала на Mandelbrot 2.


```
#include <omp.h>
# define NPOINTS 1000
# define MXITR 1000
void testpoint(void);
struct d_complex{
  double r;    double i;
};
struct d_complex c;
int numoutside = 0;

int main(){
  int i, j;
  double area, error, eps  = 1.0e-5;
#pragma omp parallel for default(shared) private(c,eps)
  for (i=0; i<NPOINTS; i++) {
    for (j=0; j<NPOINTS; j++) {
      c.r = -2.0+2.5*(double)(i)/(double)(NPOINTS)+eps;
      c.i = 1.125*(double)(j)/(double)(NPOINTS)+eps;
      testpoint();
    }
  }
area=2.0*2.5*1.125*(double)(NPOINTS*NPOINTS-
numoutside)/(double)(NPOINTS*NPOINTS);
  error=area/(double)NPOINTS;
}
```

```
void testpoint(void){
struct d_complex z;
      int iter;
      double temp;

      z=c;
      for (iter=0; iter<MXITR; iter++){
        temp = (z.r*z.r)-(z.i*z.i)+c.r;
        z.i = z.r*z.i*2+c.i;
        z.r = temp;
        if ((z.r*z.r+z.i*z.i)>4.0) {
          numoutside++;
          break;
        }
      }
}
```

Площ на фрактала на Mandelbrot 3.


```
#include <omp.h>
# define NPOINTS 1000
# define MXITR 1000
struct d_complex{
  double r;    double i;
};
void testpoint(struct d_complex);
struct d_complex c;
int numoutside = 0;

int main(){
  int i, j;
  double area, error, eps  = 1.0e-5;
#pragma omp parallel for default(shared) private(c, j) \
   firstpriivate(eps)
  for (i=0; i<NPOINTS; i++) {
    for (j=0; j<NPOINTS; j++) {
      c.r = -2.0+2.5*(double)(i)/(double)(NPOINTS)+eps;
      c.i = 1.125*(double)(j)/(double)(NPOINTS)+eps;
      testpoint(c);
    }
  }
area=2.0*2.5*1.125*(double)(NPOINTS*NPOINTS-numoutside)/(double)(NPOINTS*NPOINTS);
  error=area/(double)NPOINTS;
```

```
void testpoint(struct  d_complex c){
struct d_complex z;
      int iter;
      double temp;

      z=c;
      for (iter=0; iter<MXITR; iter++){
        temp = (z.r*z.r)-(z.i*z.i)+c.r;
        z.i = z.r*z.i*2+c.i;
        z.r = temp;
        if ((z.r*z.r+z.i*z.i)>4.0) {
        #pragma omp atomic
          numoutside++;
          break;
        }
      }
}
```

```
#include <omp.h>
static long num_steps = 100000;
double step;
void main ()
{  int i;   double x, pi, sum = 0.0;
#pragma omp parallel for private(x) reduction(+:sum)
  step = 1.0/(double) num_steps;

  for (i=0;i< num_steps; i++){
  x = (i+0.5)*step;
  sum = sum + 4.0/(1.0+x*x);
  }
  pi = step * sum;
}
```

от *Introduction to OpenMP 11 part 2 Module 6* до *Introduction to OpenMP 13 Discussion 5*