OpenMP

- Lab 2 -

Sumar

- Loop scheduling
- Reduction

Loop scheduling

#pragma omp for schedule(<mode>,[<chunk>])

Mode:

- static – default pentru gcc
- dynamic
- guided
- auto
- runtime

Loop scheduling

Mode – static, chunk=4

- munca este distribuita circular
- util pentru operatii care dureaza similar

```
schedule(static, 4):
****            ****            ****            ****
    ****            ****            ****            ****
        ****            ****            ****            ****
            ****            ****            ****            ****
```

Loop scheduling

Mode – dynamic – util pentru taskuri cu timp variabil de executie

```
schedule(dynamic, 1):
    *    *     *        *   *    * *  *  *         *  * *  * *
*  *  *   * *     *  * * *    * *      *   ***  *   *         *
 *   *  *  *  *    ** *    *      *  *  * *   *  *   *   *
  *    *     *  **        *  * *    *          *  *    *  * *  *
```

```
schedule(dynamic, 4):
            ****                    ****                    ****
****            ****    ****            ****        ****
    ****            ****    ****            ****        ****
        ****                    ****            ****
```

Loop scheduling

Mode – guided, min chunk_size work

```
schedule(guided, 2):
                ************                     ****     **
                                     *******         ***    **
                            *********
****************                            *****       **    **
```

```
schedule(guided, 4):
                                     *******
                ************                     ****    ****
                            *********
****************                            *****    ****    ***
```

Reduction

#pragma omp for reduction(+: variable)

Operatori: +, -, *, &, |, ^, &&, ||

Exemplu:

```
sum = 0;
#pragma omp parallel for shared(a) reduction(+: sum)
for (i = 0; i < 10; i++)
{
    sum += a[i]
}
```

Task-uri

1)Pornind de la ex1.c:

1. Compilati programul (gcc -fopenmp ex1.c -o ex1)
2. Rulati programul de mai multe ori. Ce observati?
3. Modificati codul schimband politica de scheduling in static. Ce observati?
4. Se observa schimbari de performanta?

Task-uri

2) Folosind Reduction, paralelizati codul in ex2.c

3) Folosind gen_files.sh, generati o lista de fisiere text. Sa sa se numere de cate ori se regasesc caracterele A-Z in continutul fisierelor. Scrieti programul serial pe care apoi il paralelizati folosind openmp. Output: A:10,B:4,...Z:100