Loop scheduling

Mehti Musayev
M4115c academic group

What is loop scheduling?

- schedule controls how loop iterations are divided among threads, i.e. is used to distribute the iterations of the loop among the threads in roughly equal amounts via the scheduling policy

- has different kinds according the task and goals

Scheduling types (kinds)

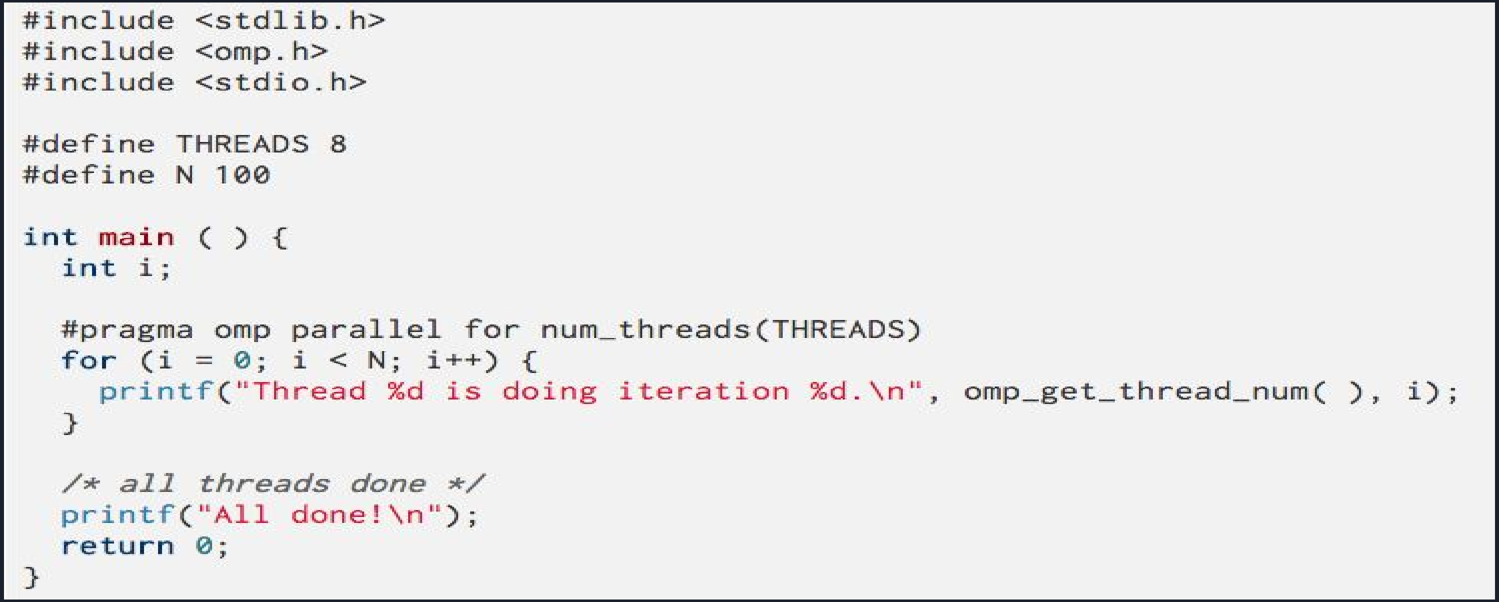
- static
- dynamic
- guided
- auto
- runtime

static type

Divide the loop into equal-sized chunks or as equal as possible in the case where the number of loop iterations is not evenly divisible by the number of threads multiplied by the chunk size. By default, chunk size is loop_count/number_of_threads.

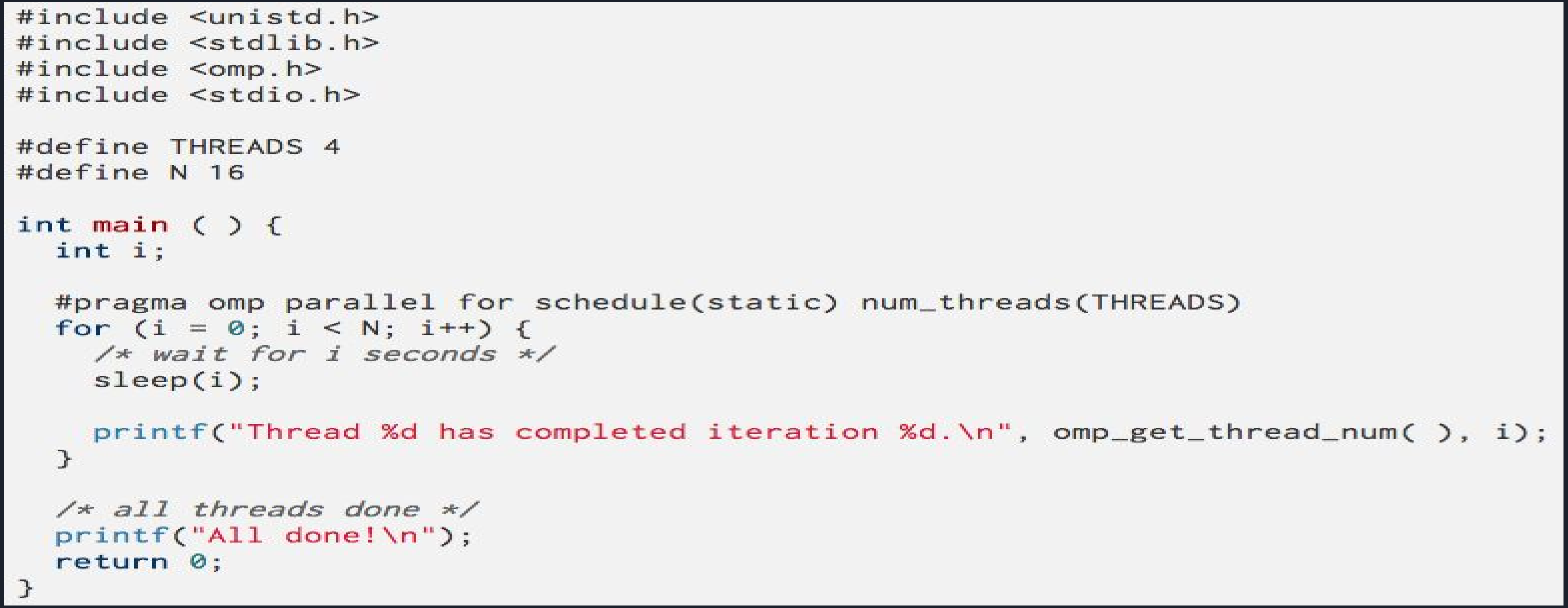
Set chunk to 1 to interleave the iterations.

static type example



```c
#include <stdlib.h>
#include <omp.h>
#include <stdio.h>

#define THREADS 8
#define N 100

int main ( ) {
  int i;

  #pragma omp parallel for num_threads(THREADS)
  for (i = 0; i < N; i++) {
    printf("Thread %d is doing iteration %d.\n", omp_get_thread_num( ), i);
  }

  /* all threads done */
  printf("All done!\n");
  return 0;
}
```

static type example



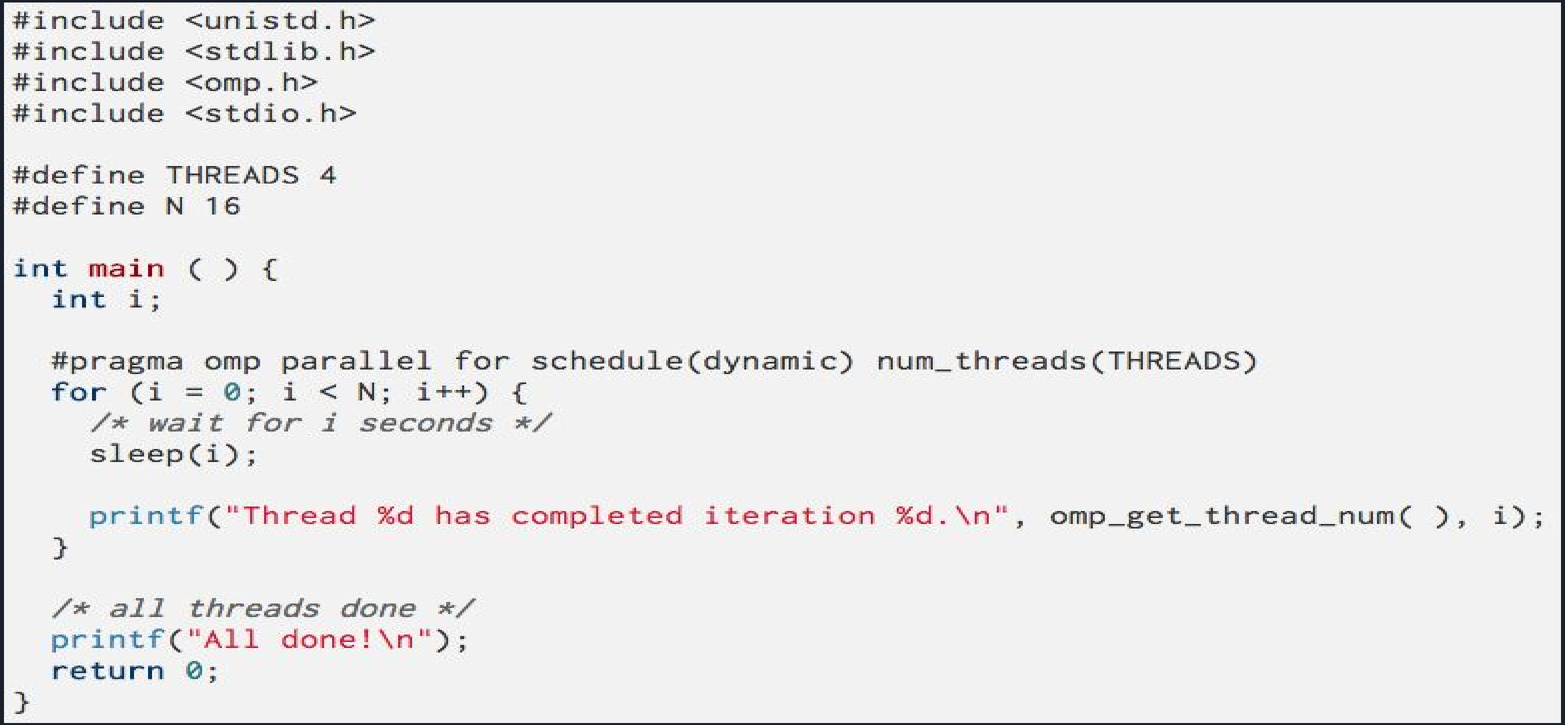
```c
#include <unistd.h>
#include <stdlib.h>
#include <omp.h>
#include <stdio.h>

#define THREADS 4
#define N 16

int main ( ) {
  int i;

  #pragma omp parallel for schedule(static) num_threads(THREADS)
  for (i = 0; i < N; i++) {
    /* wait for i seconds */
    sleep(i);

    printf("Thread %d has completed iteration %d.\n", omp_get_thread_num( ), i);
  }

  /* all threads done */
  printf("All done!\n");
  return 0;
}
```

dynamic type

Use the internal work queue to give a chunk-sized block of loop iterations to each thread. When a thread is finished, it retrieves the next block of loop iterations from the top of the work queue.

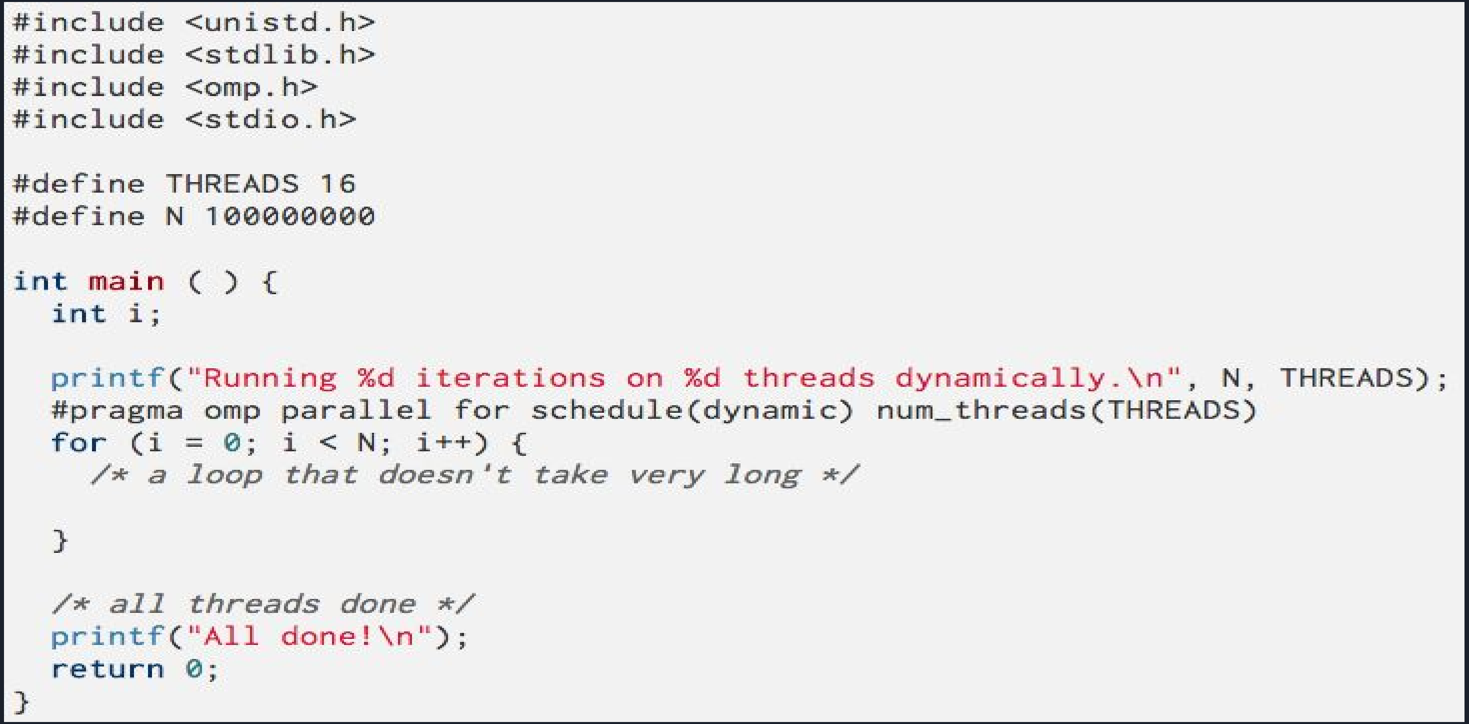
By default, the chunk size is 1. Be careful when using this scheduling type because of the extra overhead involved.

dynamic type example



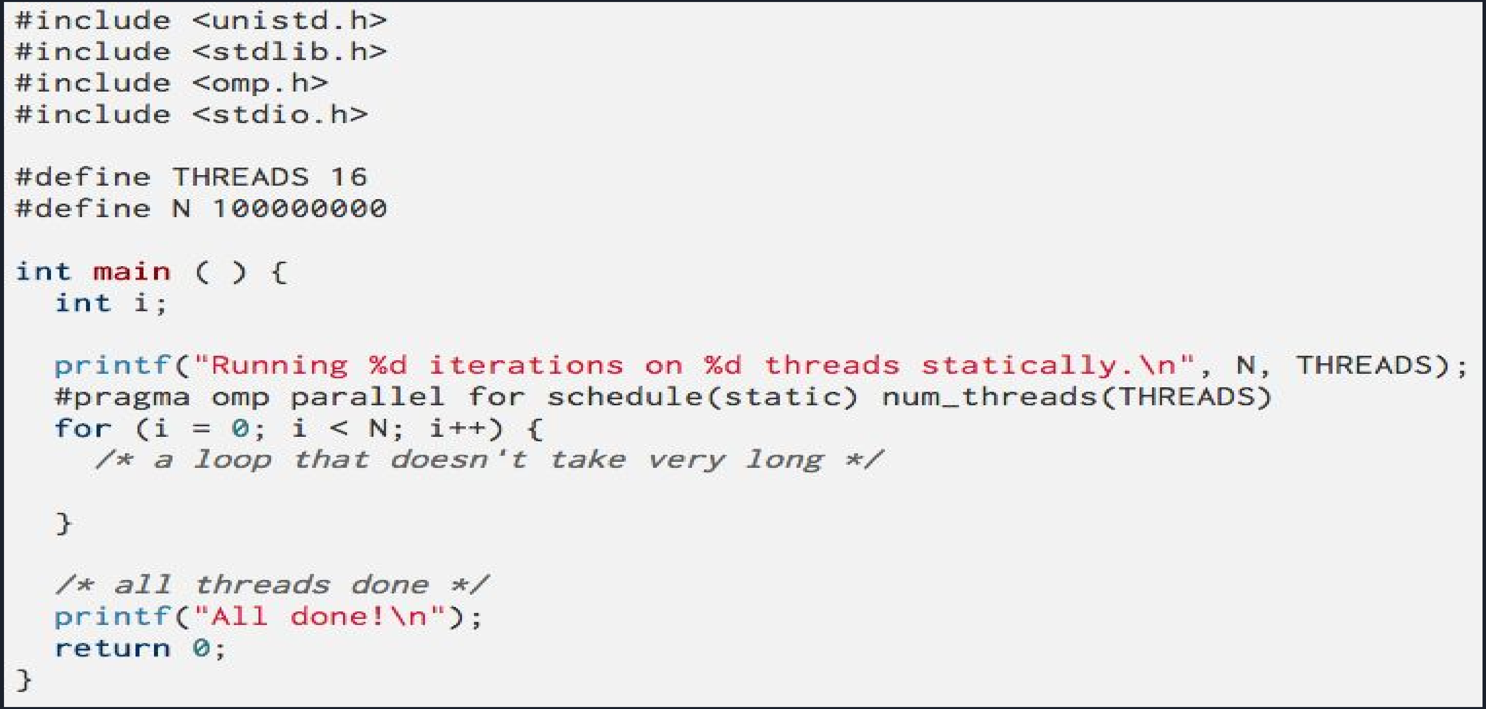
```c
#include <unistd.h>
#include <stdlib.h>
#include <omp.h>
#include <stdio.h>

#define THREADS 4
#define N 16

int main ( ) {
  int i;

  #pragma omp parallel for schedule(dynamic) num_threads(THREADS)
  for (i = 0; i < N; i++) {
    /* wait for i seconds */
    sleep(i);

    printf("Thread %d has completed iteration %d.\n", omp_get_thread_num( ), i);
  }

  /* all threads done */
  printf("All done!\n");
  return 0;
}
```

dynamic type example (overhead)



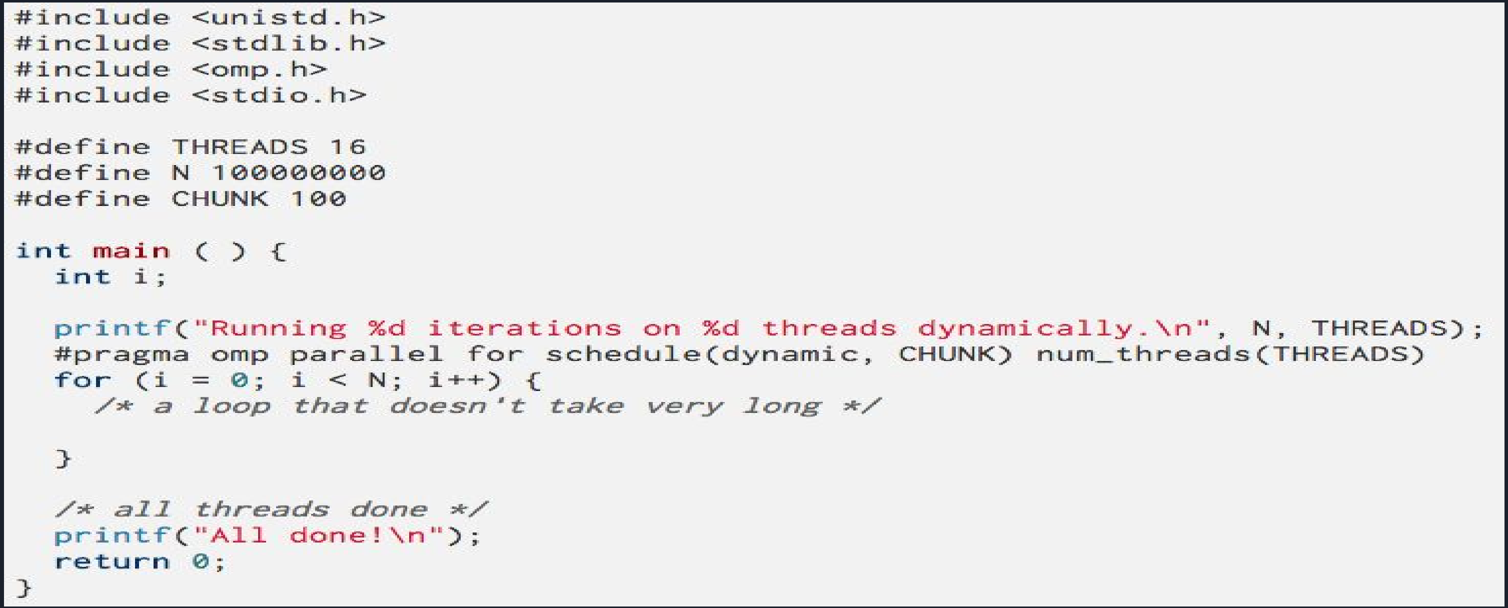
```
#include <unistd.h>
#include <stdlib.h>
#include <omp.h>
#include <stdio.h>

#define THREADS 16
#define N 100000000

int main ( ) {
  int i;

  printf("Running %d iterations on %d threads dynamically.\n", N, THREADS);
  #pragma omp parallel for schedule(dynamic) num_threads(THREADS)
  for (i = 0; i < N; i++) {
    /* a loop that doesn't take very long */

  }

  /* all threads done */
  printf("All done!\n");
  return 0;
}
```

dynamic type example (fixing overhead)



```c
#include <unistd.h>
#include <stdlib.h>
#include <omp.h>
#include <stdio.h>

#define THREADS 16
#define N 100000000

int main ( ) {
  int i;

  printf("Running %d iterations on %d threads statically.\n", N, THREADS);
  #pragma omp parallel for schedule(static) num_threads(THREADS)
  for (i = 0; i < N; i++) {
    /* a loop that doesn't take very long */

  }

  /* all threads done */
  printf("All done!\n");
  return 0;
}
```

chunk sizes in static and dynamic scheduling



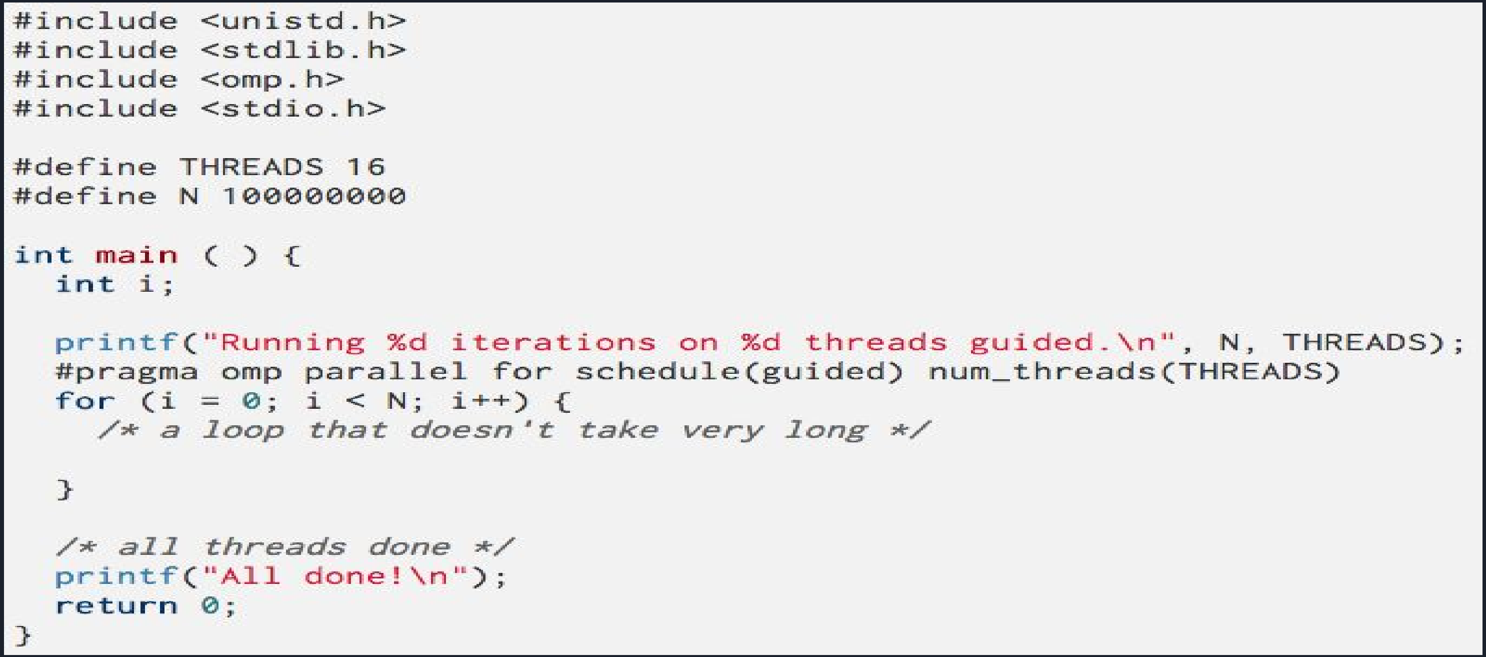
```c
#include <unistd.h>
#include <stdlib.h>
#include <omp.h>
#include <stdio.h>

#define THREADS 16
#define N 100000000
#define CHUNK 100

int main ( ) {
  int i;

  printf("Running %d iterations on %d threads dynamically.\n", N, THREADS);
  #pragma omp parallel for schedule(dynamic, CHUNK) num_threads(THREADS)
  for (i = 0; i < N; i++) {
    /* a loop that doesn't take very long */

  }

  /* all threads done */
  printf("All done!\n");
  return 0;
}
```

guided type

Similar to dynamic scheduling, but the chunk size starts off large and decreases to better handle load imbalance between iterations. The optional chunk parameter specifies them minimum size chunk to use.

By default the chunk size is approximately loop_count/number_of_threads.

guided type example



```c
#include <unistd.h>
#include <stdlib.h>
#include <omp.h>
#include <stdio.h>

#define THREADS 16
#define N 100000000

int main ( ) {
  int i;

  printf("Running %d iterations on %d threads guided.\n", N, THREADS);
  #pragma omp parallel for schedule(guided) num_threads(THREADS)
  for (i = 0; i < N; i++) {
    /* a loop that doesn't take very long */

  }

  /* all threads done */
  printf("All done!\n");
  return 0;
}
```

auto type

When schedule (auto) is specified, the decision regarding scheduling is delegated to the compiler. The programmer gives the compiler the freedom to choose any possible mapping of iterations to threads in the team.

runtime type

Uses the OMP_SCHEDULE environment variable to specify which one of the three loop-scheduling types should be used.

OMP_SCHEDULE is a string formatted exactly the same as would appear on the parallel construct.

References

- https://software.intel.com/en-us/node/522683
- http://cs.umw.edu/~finlayson/class/fall16/cpsc425/notes/12-scheduling.html
- https://stackoverflow.com/questions/10850155/openmp-for-schedule
- OMP_SCHEDULE environment: https://software.intel.com/en-us/node/522775#CC142A61-48BB-48A0-8062-92A4F266B6BE

Thank you!

- email: mehty.musaev@gmail.com
- c.p.: +7-911-039-51-35
- LinkedIn: https://www.linkedin.com/in/mehtimusayev/
- presentation: https://github.com/b5y/parallel_programing