Shared Memory Programming

More about parallel loops

LASTPRIVATE clause

- Sometimes need the value a private variable would have had on exit from loop (normally undefined).

Syntax:

Fortran: `LASTPRIVATE(list)`

C/C++: `lastprivate(list)`

- Also applies to *sections* directive (variable has value assigned to it in the last section.)

LASTPRIVATE clause (cont)

Example:

```
!$OMP PARALLEL
!$OMP DO LASTPRIVATE(i)
      do i=1,func(l,m,n)
         d(i)=d(i)+e*f(i)
      end do
      ix = i-1
      . . .
!$OMP END PARALLEL
```

SCHEDULE clause

- The SCHEDULE clause gives a variety of options for specifying which loops iterations are executed by which thread.
- Syntax:

Fortran: **`SCHEDULE (`***kind[, chunksize]*`)`**

C/C++: **`schedule (`***kind[, chunksize]*`)`**

where *kind* is one of

`STATIC`, **`DYNAMIC`**, **`GUIDED`** or **`RUNTIME`**

and *chunksize* is an integer expression with positive value.

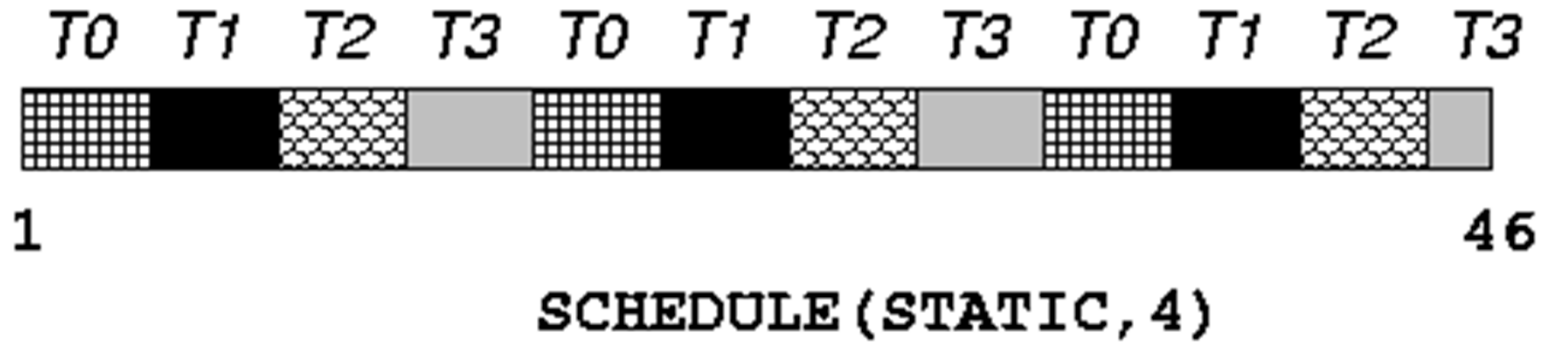
- E.g. **`!$OMP DO SCHEDULE(DYNAMIC,4)`**

STATIC schedule

- With no *chunksize* specified, the iteration space is divided into (approximately) equal chunks, and one chunk is assigned to each thread in order (**block** schedule).

- If *chunksize* is specified, the iteration space is divided into chunks, each of *chunksize* iterations, and the chunks are assigned cyclically to each thread in order (**block cyclic** schedule)

STATIC schedule



archer
epcc
THE UNIVERSITY OF EDINBURGH

DYNAMIC schedule

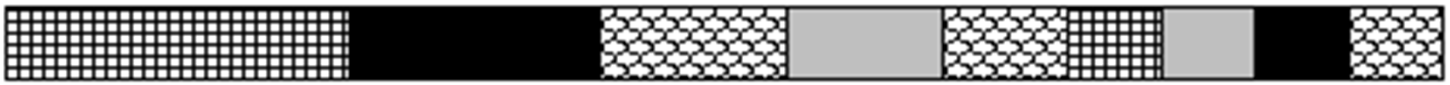
- DYNAMIC schedule divides the iteration space up into chunks of size *chunksize*, and assigns them to threads on a first-come-first-served basis.
- i.e. as a thread finish a chunk, it is assigned the next chunk in the list.
- When no *chunksize* is specified, it defaults to 1.

GUIDED schedule

- GUIDED schedule is similar to DYNAMIC, but the chunks start off large and get smaller exponentially.
- The size of the next chunk is proportional to the number of remaining iterations divided by the number of threads.
- The *chunksize* specifies the minimum size of the chunks.
- When no *chunksize* is specified it defaults to 1.

DYNAMIC and GUIDED schedules

1 SCHEDULE(DYNAMIC,3) 46



1 SCHEDULE(GUIDED,3) 46

archer
epcc
THE UNIVERSITY OF EDINBURGH

RUNTIME schedule

- The RUNTIME schedule defers the choice of schedule to run time, when it is determined by the value of the environment variable **`OMP_SCHEDULE`**.

- e.g. `export OMP_SCHEDULE="guided,4"`

- It is illegal to specify a chunksize in the code with the RUNTIME schedule.

Choosing a schedule

When to use which schedule?

- STATIC best for load balanced loops - least overhead.
- STATIC,*n* good for loops with mild or smooth load imbalance, but can induce overheads.
- DYNAMIC useful if iterations have widely varying loads, but ruins data locality.
- GUIDED often less expensive than DYNAMIC, but beware of loops where the first iterations are the most expensive!
- Use RUNTIME for convenient experimentation.

ORDERED directive

- Can specify code within a loop which must be done in the order it would be done if executed sequentially.
- Syntax:

Fortran:
```
!$OMP ORDERED
        block
!$OMP END ORDERED
```

C/C++:
```
#pragma omp ordered
        structured block
```

- Can only appear inside a DO/FOR directive which has the ORDERED clause specified.
- Main use is in testing to force ordering of output

ORDERED directive (cont)

Example:

```
!$OMP PARALLEL DO ORDERED
      do j=1,n
         . . .
!$OMP ORDERED
          write(*,*) j,count(j)
!$OMP END ORDERED
      . . .
      end do
!$OMP END PARALLEL DO
```

Practical session

Finding Goldbach pairs

- Aim: experiment with loop schedules.
- The Goldbach conjecture says that every even number greater than 2 is the sum of 2 primes.
- For the first 4000 even numbers, find all pairs of primes which sum to the even number.
- Computational cost rises as $n^{3/2}$, giving an unbalanced load
- Parallelise with a DO directive, and experiment with different schedule options.