Lecture 15: Parallel Regions and Loops in OpenMP

CMSE 822: Parallel Computing

Prof. Sean M. Couch

VOTE!

- michigan.gov/vote
- https://www.betterknowaballot.com
- Mail absentee ballot TODAY! (or better yet, drop it off at clerk's office)

Puppy time!

PCA Questions

Emily Bolger 1:21 PM

PCA13: Can you explain more what guided scheduling looks like and when it would be useful?

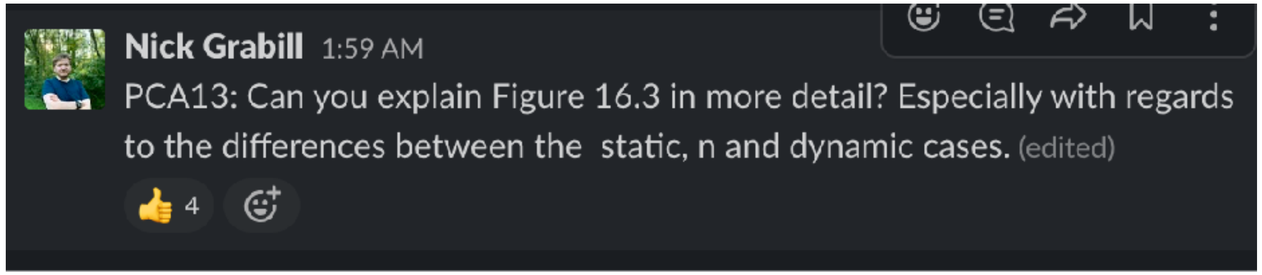
👍 4

PCA Questions

Kind	Description
static	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.
dynamic	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.
guided	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.
auto	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	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.

PCA Questions

Scheduling



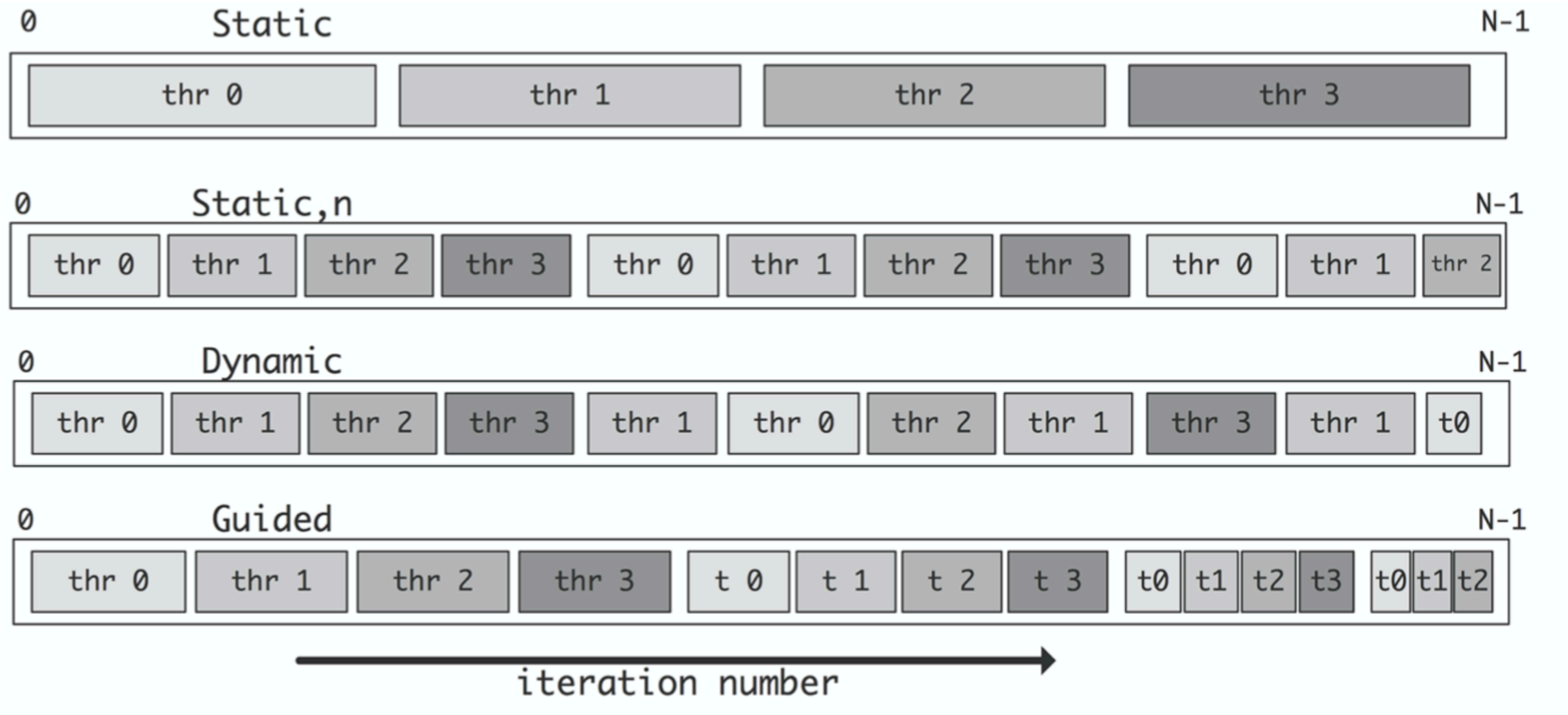
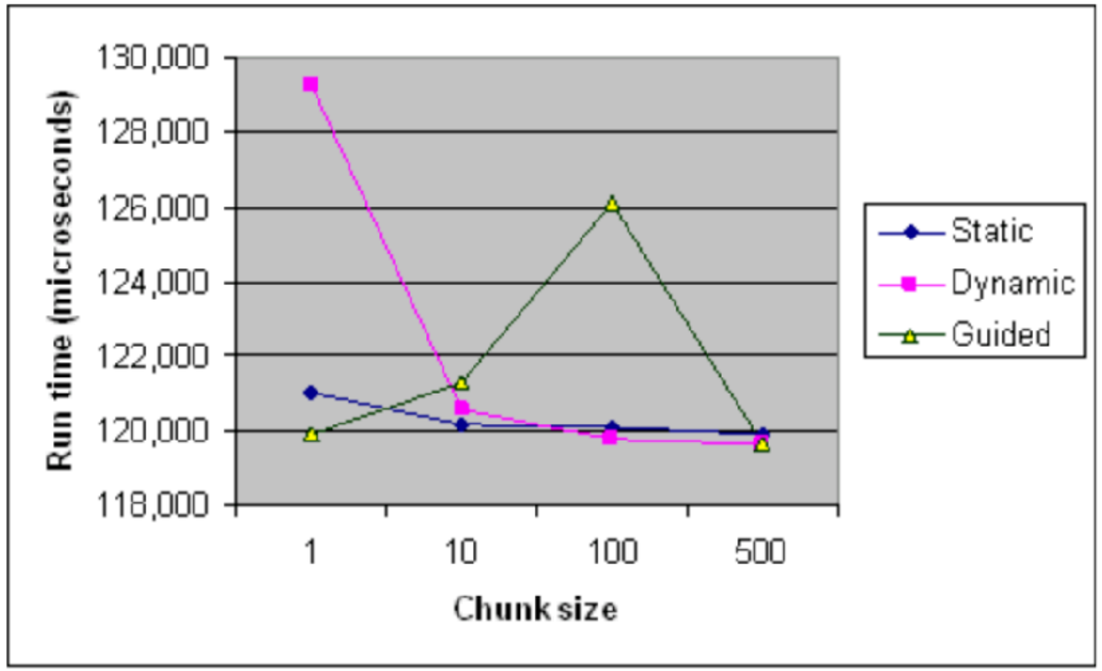


Figure 16.3: Illustration of the scheduling strategies of loop iterations

PCA Questions

Scheduling

- Gets complicated! See, e.g., https://stackoverflow.com/questions/42970700/openmp-dynamic-vs-guided-scheduling



PCA Questions

Scheduling

- Goal is good **load balance** with low **overhead**.
- Can be implementation (i.e., compiler) specific
- Guided and dynamic have larger overheads (guided the most?)
- Cache and NUMA effects can play a role..

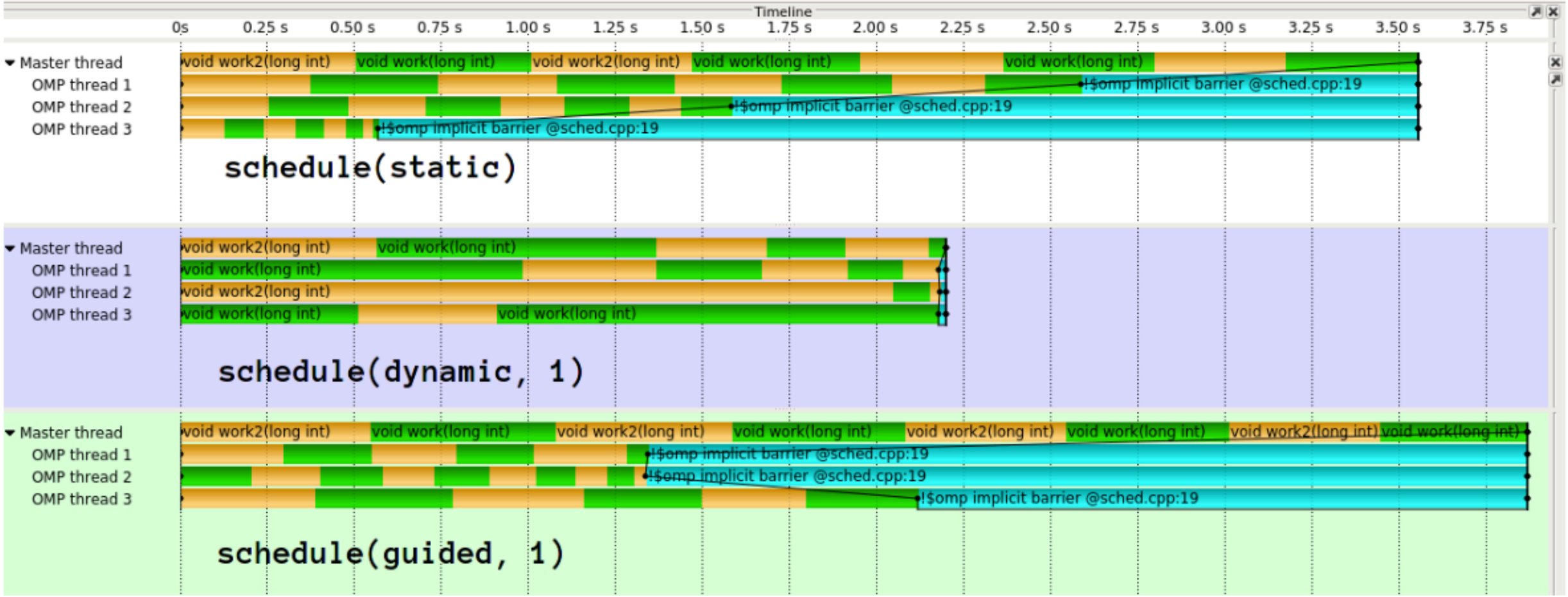
```
#include <omp.h>

void work(long ww) {
    volatile long sum = 0;
    for (long w = 0; w < ww; w++) sum += w;
}

int main() {
    const long max = 32, factor = 10000000l;
    #pragma omp parallel for schedule(guided, 1)
    for (int i = 0; i < max; i++) {
        work((max - i) * factor);
    }
}
```

PCA Questions

Scheduling



PCA Questions

Matthew Zeilbeck 12:08 AM

PCA13: What does the `auto` scheduler actually do? The book says that it's left up to the implementation. Does that mean that the specific OpenMP implementation on the machine has a built-in default schedule that is invoked?

Also, in the code examples, in the `#pragma` statement, the schedule is specified with something like `schedule(static,...)`, but the book also says in the bulleted list on page 408 to set by using value `omp_sched_static`. Is the latter for use if we set the `runtime` schedule with `omp_set_schedule`? Is it an example of an `omp_sched_t` object?

👍 7

PCA Questions

Brandon Barker 8:51 PM

PCA13 -- we see that `omp for` can only do a single loop without using `collapse(2)`. (16.4) However, this can only work for "perfectly nested" loops. We saw before tricks for nested loops to keep variables in register. It seems that for this, we could have to abandon such possible tricks. In a scenario where such tricks are possible, how might we choose which optimization to make -- force variables to stay in register or use OMP with both loops? (edited)

👍 1

PCA Questions

Cache blocking in OpenMP

- https://moodle.rrze.uni-erlangen.de/pluginfile.php/10881/mod_resource/content/3/05_Roofline_Jacobi.pdf

```
do jb=1,jmax,jblock !  Assume jmax is multiple of jblock

!$OMP PARALLEL DO SCHEDULE(STATIC)
  do k=1,kmax
    do j=jb, (jb+jblock-1) !  Loop length jblock
      do i=1,imax
        y(i,j,k) = 1/6. *(x(i-1,j,k) +x(i+1,j,k) &
                        +    x(i,j-1,k) +x(i,j+1,k)
                        +    x(i,j,k-1) +x(i,j,k+1))
      enddo
    enddo
  enddo
enddo
```

"Layer condition" (j-Blocking)

nthreads*3***jblock*****imax***8B < CS/2

Ensure layer condition by choosing **jblock** appropriately (cubic domain):

jblock < CS/(imax* nthreads* 48B **)**

Group work: HW8!