You can find the slides at:
https://drive.google.com/file/d/1nVhI8Ng70vGcnzcToAxBiiF0Tuht1xr5/view?usp=sharing

Some notes on the slides:

1. Slides 2-8, especially slides 3-8 were included as background for science majors, who often need a physical intuition into the problem to understand a code. They are the same as can be found in the parallel lessons in the MPI and OpenACC units. An instructor should feel free to provide them as background, or to use them as they see fit.
2. The rest of the slides are a walk through parallelizing the code.
3. A MATLAB script (NBodyMovie.m) is included if a student would like to visualize the output.
4. Sample Student Assessment:
    a. For problem 1, the student only needs to change the line "int Nbodies=10000;" to a new larger number.
    b. For problems 2 & 3 the student needs to modify the "#pragma omp parallel for" to add a schedule specifier (e.g. "#pragma omp parallel for simd private(vx,vy)" to "#pragma omp parallel for simd private(vx,vy) schedule(static,4)" to set the scheduling to static with a chunk size of 4. In addition to trying different chunk sizes, students should try different scheduling types (static, dynamic, guided, auto, runtime) to see how the runtime is affected. Why are some better than others (a good hint might be to tell them to think about to the single processor optimization unit [if you used it] and to think about cache size)
5. Sample results, scheduling for acceleration module (Quad Core Intel Xeon CPU W3565 @ 3.20GHz, 4 threads, PGI compiler 19.10) -- your student's results may vary:

Scheduling Type	Run time (ms)
Default	6.9×10^7
static,2	1.3×10^8
static,4	1.2×10^8
static,8	8.7×10^7
static, 16	7.8×10^7
static, 32	7.1×10^7

static, 64	$7.0x10^7$
static,128	$7.0x10^7$
static, 256	$7.1x10^7$
dynamic, 16	$7.0x10^7$
dynamic, 32	$6.9x10^7$
dynamic, 64	$7.0x10^7$
dynamic, 128	$7.1x10^7$
auto	$7.1x10^7$
guided	$7.1x10^7$

If we hold the acceleration loop at dynamic,32 and look at the position update loops:

Scheduling type	Runtime (ms)
static,16	$6.9x10^7$
static,32	$6.9x10^7$
static,64	$7.0x10^7$
static,128	$7.0x10^7$
dynamic,16	$6.9x10^7$
dynamic,32	$7.0x10^7$
dynamic,64	$6.9x10^7$
dynamic,128	$7.0x10^7$
auto	$7.0x10^7$
guided	$7.0x10^7$

Holding the acceleration and position update loops at dynamic,32 and just changing the position initialization loop

Scheduling type	Time to init (ms)
static,16	6.93×10^6
static,32	6.95×10^6
static,64	6.92×10^6
static,128	7.10×10^6
dynamic,16	6.92×10^6
dynamic,32	6.93×10^6
dynamic,64	6.95×10^6
dynamic,128	7.04×10^6
auto	6.91×10^6
guided	6.90×10^6

The heavy math loops seem to work best with a chunk size greater than 32 (likely due to cache usage), the initialization loop works best at low chunk size, or guided -- why? The initialization loop is unique in that each iteration of the outer loop may take a different amount of time than the other loops -- thus small chunks and guided (which starts large and gets smaller) produce better load balancing.