Lab #4: OpenMP parallel for/do loops

PCSE 2015

Lab assignment

The file `pi11.c` computes π by Riemann integration:

$$\pi = 4\int_0^1 y\,dx \quad \text{where } x^2+y^2=1$$

The difference with the previous lab is that this code tries to use 'adaptive integration': where needed it refines the step size[1]. This means that the iterations no longer take a predictable amount of time.

```
for (i=0; i<nsteps; i++) {
  double
    x = i*h,x2 = (i+1)*h,
    y = sqrt(1-x*x),y2 = sqrt(1-x2*x2),
    slope = (y-y2)/h;
  if (slope>15) slope = 15;
  int
    samples = 1+(int)slope,
    is;
  for (is=0; is<samples; is++) {
    double
      hs = h/samples,
      xs = x+ is*hs,
      ys = sqrt(1-xs*xs);
    quarterpi += hs*ys;
    nsamples++;
  }
}
pi = 4*quarterpi;
```

1. It doesn't actually do this in a mathematically sophisticated way, so this code is more for the sake of the example.

1. Use the `omp parallel for` construct to parallelize the loop. As in the previous lab, you may at first see an incorrect result. Use the `reduction` clause to fix this.
2. Your code should now see a decent speedup, using up to 8 cores. However, it is possible to get completely linear speedup. For this you need to adjust the schedule.
   Start by using `schedule(static,`n`)`. Experiment with values for n. When can you get a better speedup? Explain this.
3. Since this code is somewhat dynamic, try `schedule(dynamic)`. This will actually give a fairly bad result. Why? Use `schedule(dynamic,`n`)` instead, and experiment with values for n.
4. Finally, use `schedule(guided)`, where OpenMP uses a heuristic. What results does that give?