More OpenMP

- `schedule` clause
- `collapse` clause (OpenMP 3.0)
- Matrix power example
Specifying Data Distributions

Using just

```
#pragma omp for
```

leaves the decision of data allocation up to the compiler

When you want to specify it yourself, use `schedule`:

```
#pragma omp for schedule(....)
```

See color Mandelbrot example...
Mandelbrot: Speed-up with 4 Processors

Sequential performance:

4000 pixels: 2.59 seconds, 99% CPU

Measurements use a Xeon 4-CPU machine, gcc 4.1.2, and Linux 2.6.23
Mandelbrot: Speed-up with 4 Processors

4 processors, \textit{schedule\,(auto)}

01.04 seconds, 239\% CPU

256 pixels

2000 pixels
Mandelbrot: Speed-up with 4 Processors

4 processors, \( \text{schedule}(\text{static}) \)

0.93 seconds, 267% CPU

256 pixels  

2000 pixels
Mandelbrot: Speed-up with 4 Processors

4 processors, \texttt{schedule(static, 1)}

00.86 seconds, 289% CPU

256 pixels  2000 pixels
Mandelbrot: Speed-up with 4 Processors

4 processors, $\text{schedule}(\text{static}, 10)$

00.85 seconds, 294% CPU

256 pixels  
2000 pixels
Mandelbrot: Speed-up with 4 Processors

4 processors, \texttt{schedule(dynamic)}

00.72 seconds, 350\% CPU

256 pixels  2000 pixels
Mandelbrot: Speed-up with 4 Processors

4 processors, \texttt{schedule(dynamic, 10)}

00.66 seconds, 381\% CPU
Mandelbrot: Speed-up with 4 Processors

4 processors, \textit{schedule} (guided)

00.83 seconds, 302% CPU

256 pixels

2000 pixels
Mandelbrot: Speed-up with 4 Processors

4 processors, \( \text{schedule}(\text{guided}, \ 10) \)

00.78 seconds, 318% CPU

256 pixels

2000 pixels
Collapsing Loops

Suppose that we want to break up the Mandelbot image in finer granularities than a line:

```c
int y; ...
#pragma omp parallel for
for (y = 0; y < h; ++y) {
  int x; ...
  for (x = 0; x < w; ++x) { ... }
}
```

With OpenMP 3.0 (not supported on CADE installation), you could use the `collapse` clause

```c
int x, y; ...
#pragma omp parallel for collapse(2)
for (y = 0; y < h; ++y)
  for (x = 0; x < w; ++x) { ... }
```