21st Summer School of PARALLEL COMPUTING

July 2 - 13, 2012 (Italian)
September 3 - 14, 2012 (English)

OpenMP Exercises

A tutorial lab in three steps

Massimiliano Culpo[1], Gianfranco Marras[1]

[1]CINECA - SuperComputing Applications and Innovation Department

CINECA

Warm-up with OpenMP

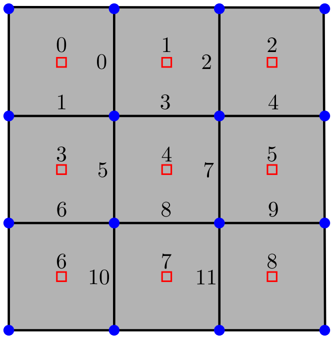
1. Write a serial "Hello world!"
2. Add OpenMP directives to have each thread prompt his greeting
3. Add a conditionally compiled header to show if OpenMP was enabled
4. Experiment with the OMP_NUM_THREADS environment variable
5. Implement a routine that returns
   - thread ID
   - number of threads

   without using OpenMP library calls
6. Have each thread print his ID and the total number of threads in an *ordered* fashion

Smooth

1. Parallelize the serial code acting only on the most important loop
   - use *default(none)*
   - try different strategies to write on shared variables
   - try different schedules and test their performance
2. Parallelize the serial code with a single fork at the beginning of main()
   - use *default(none)*
   - proceed incrementally
   - remember data-copying clauses

LDU Matrix Format: Data Structure



Storage Format Quick Guide

1. Matrix is considered
   - square
   - structurally symmetric
   - sum of three parts ($A = L + D + U$)
2. Off-diagonal positions mapping
   - `lPtr` (globally ordered)
   - `uPtr` (locally ordered)
3. Values stored in three double vectors
4. No fill-in introduced

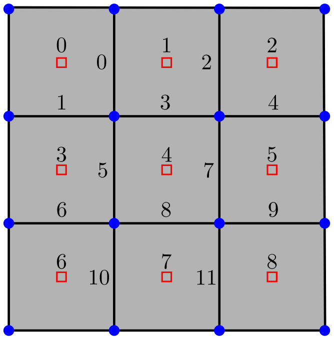
$$\texttt{lPtr} = \begin{bmatrix} 0 & 0 & 1 & 1 & 2 & 3 & 3 & 4 & 4 & 5 & 6 & 7 \end{bmatrix}$$
$$\texttt{uPtr} = \begin{bmatrix} 1 & 3 & 2 & 4 & 5 & 4 & 6 & 5 & 7 & 8 & 7 & 8 \end{bmatrix}$$

Serial Matrix-Vector Multiplication

```
// Diagonal contributions
for (int cell=0; cell < nCells; cell++) {
   ApsiPtr[cell] = diagPtr[cell]*psiPtr[cell];
}

// Off-diagonal contributions
for (int face=0; face < nFaces; face++) {
   ApsiPtr[uPtr[face]] += lowerPtr[face]*psiPtr[lPtr[face]];
   ApsiPtr[lPtr[face]] += upperPtr[face]*psiPtr[uPtr[face]];
}
```

LDU Matrix Format: Modifications



What prevents multi-threading?

1. Off-diagonal contributions
   - Concurrent write-access
   - Access by cells needed
2. Owner sort
   - `owPtr`
3. Losort
   - `reshape`
   - `loPtr`
4. Introduce doubly indirect access of rvalues

$$\texttt{owPtr} = \begin{bmatrix} 0 & 2 & 4 & 5 & 7 & 9 & 10 & 11 & 12 & 12 \end{bmatrix}$$

$$\texttt{loPtr} = \begin{bmatrix} 0 & 0 & 1 & 2 & 3 & 5 & 7 & 8 & 10 & 12 \end{bmatrix}$$

$$\texttt{reshape} = \begin{bmatrix} 0 & 2 & 1 & 3 & 5 & 4 & 7 & 6 & 8 & 10 & 9 & 11 \end{bmatrix}$$

OpenMP Matrix-Vector Multiplication

```
// Diagonal contributions
#pragma omp for
for (int cell=0; cell < nCells; cell++) {
   ApsiPtr[cell] = diagPtr[cell]*psiPtr[cell];
}

// Off-diagonal contributions
#pragma omp for
for (int cell=0; cell < nCells; ++cell) {
   for (int fidx = owPtr[cell]; fidx < owPtr[cell+1]; ++fidx)
      AxPtr[cell] += upperPtr[fidx]*xPtr[uPtr[fidx]];
   for (int fidx = loPtr[cell]; fidx < loPtr[cell+1]; ++fidx)
      AxPtr[cell] += lowerPtr[reshape[fidx]]*
                     xPtr[lPtr[reshape[fidx]]];
}
```