Advanced OpenMP: Hands-on Session

Christian Terboven

Michael Klemm

Xavier Teruel

Bronis R. de Supinski

Members of the OpenMP Language Committee

01a - Using tasks (sudoku)

- Sudoku is a popular Japanese puzzle game based on the placement of numbers on a square board. For each position in the board the algorithm tries each possible combination.
- Source file structure
  - → SudokuBoard.cpp, SudokuBoard.h → Sudoku class definition
  - → sudoku.cpp → contains the main program and solver. Candidate to parallelize.
- Exercise goals
  - → Focus on the annotated TODO's spread among the code (at sudoku.cpp)
  - → Create the parallel region to guarantee a single creator, multiple executors
  - → Create tasks when required. Add proper synchronization mechanisms.
  - → Discuss about the need of having two different versions of the sudoku solve function

01b - Using tasks (cholesky)

- Cholesky kernel is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose. The algorithm uses 4 MKL services to compute the final result: `gemm`, `potrf`, `trsm` and `syrk`.
- Source file structure (single file)
  - → cholesky.c → contains the main program and Cholesky solver
- Exercise goals
  - → Focus on the annotated TODO's spread among the code (at cholesky.c)
  - → Create the parallel region to guarantee a single creator, multiple executors
  - → Create tasks when required. Add proper synchronization mechanisms

02a - Using task dependencies (cholesky)

- Cholesky kernel is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose. The algorithm uses 4 MKL services to compute the final result: gemm, potrf, trsm and syrk.
- Source file structure (single file)
  - → cholesky.c → contains the main program and Cholesky solver
- Exercise goals
  - → Focus on the annotated TODO's spread among the code (at cholesky.c)
  - → Taking as the starting point the previous parallelized version of Cholesky, relax the synchronization mechanisms in order to use task dependencies

03a - Using cut-off (sudoku)

- Sudoku is a popular Japanese puzzle game based on the placement of numbers on a square board. For each position in the board the algorithm tries each possible combination.
- Source file structure
  - → SudokuBoard.cpp, SudokuBoard.h → Sudoku class definition
  - → sudoku.cpp → contains the main program and solver. Candidate to parallelize.
- Exercise goals
  - → Focus on the annotated TODO's spread among the code (at sudoku.cpp)
  - → This time we will have just a single sudoku "solve" function (and not a parallel and sequential versions), add the proper cut-off mechanism to guarantee enough task granularity

03- Using cut-off (merge-sort)

- The merge-sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves.
- Source file structure (single file)
  - → mergesort.cpp → contains the main program, the sorting and merge functions
- Exercise goals
  - → Focus on the annotated TODO's spread among the code (at mergesort.cpp)
  - → Create the parallel region to guarantee a single creator, multiple executors
  - → Create tasks when required. Add proper synchronization mechanisms
  - → Add the proper cut-off mechanism to guarantee enough task granularity

04- Using cancellation (tree-search)

- A tree search algorithm attempts to find a solution by traversing a tree structure. Multiple solutions (eg, occurrences) may exist. Once one of this solutions have been found, the program may finalize.
- Source file structure (single file)
  - → treesearch.c → contains the main program, and all tree related functions
- Exercise goals
  - → Focus on the annotated TODO's spread among the code (at mergesort.cpp)
  - → Create the parallel region to guarantee a single creator, multiple executors
  - → Create tasks when required. Add proper synchronization mechanisms
  - → Add the proper cancellation scope, and cancellation points
  - → Set OMP_CANCELLATION=true in the shell