

HC-OpenMP-Conversion-Guide

Converting openmp programs to Habanero-C

Creating a task

`#pragma omp task => async`

The block of code enclosed by an omp task is now enclosed in an async block

Passing data to a task

OpenMP provide the following clause to define how a task should access variables

- `shared`: variable are shared between threads (default, except for loop iteration index)
- `private`: each thread has a local copy and use it as a temporary variable. A private variable is not initialized.
- `firstprivate`: like private except initialized to original value

Equivalent in HC:

- `shared`: The only way to share a variable between asyncs is to pass a pointer to a variable as an IN parameter. Keep in mind that the variable you are trying to share cannot be stack allocated in HC.
- `private`: The closest semantic in HC is to declare a variable at the beginning of the async body. Each async is going to have storage for the variable and is not initialized.
- `firstprivate`: Matches the semantic of the IN clause. Each async has a local copy of each IN clauses arguments and is initialized to its original value.

Waiting for tasks

`#pragma omp taskwait`

Need to create a finish block that encloses each asyncs (omp tasks) the application needs to wait for.

Atomic and Critical sections

`#pragma omp critical`: enclosed block executed by only one thread at a time

HC doesn't provide the "isolated" construct as HJ does. Currently, you'll need to use any lock implementation available in C.

One solution is to use `pthread_mutex_lock` / `pthread_mutex_unlock`

`#pragma omp atomic`: atomic operation translated to hardware specific operation.

You should use one of the primitive provided by HC. Have a look at `hc_sysdep.h` in the HC runtime src folder. It provides implementations such as atomic CAS, atomic increment, etc... for various platforms.