OpenMP's `#pragma omp task` directive allows for the creation of unitary independent tasks that can be executed by any available thread inside a parallel region.
Due to the divide-and-conquer approach of the quicksort algorithm, this directive can be directly exploited to obtain a fairly simple parallelization of this algorithm. The main idea is to have, at each recursive call, one thread that, after having partitioned the array, creates two tasks: one for the $\mathbf{X}_{<\tau}$ sub-array and one for the $\mathbf{X}_{\geq\tau}$ sub-array. Since the two sub-array formed at each step do not have any intersection, any thread that enters a task can work independently on its sub-array without the risk of contention with other threads.
This simple implementation can be schematized by the following OpenMP pseudocode[1].

OpenMP Task Quicksort

```
function omp_task_qsort(X):
    if len(X) > 2:
        τ ← choose_pivot(X)
        X_<τ, X_≥τ ← partition(X, τ)

        #pragma omp task
        omp_task_qsort(X_<τ)

        #pragma omp task
        omp_task_qsort(X_≥τ)
    else:
        if len(X) == 2 and X[0] > X[1]:
            swap(X[0], X[1])
```

Code 1: Pseudocode for the OpenMP `omp_task_qsort` function. Recursion ends when the array has length less then 2, where a simple swap is performed if needed.

The time complexity of this algorithm is the same as the serial quicksort, i.e. $\mathcal{O}(n \log n)$, but the parallelization allows the program to execute faster at a large scale. The main advantages of this approach are the simplicity of its implementation and the fact that it does not require any explicit synchronization between threads. However, we might take into account that the number of tasks created at each step can be very large, especially for large arrays, and this can lead to a significant overhead in terms of memory and time.

[1]Pseudocodes here presented don't constitute implementation examples, but only aim to give a minimal and concise idea of the main steps performed in each function by omitting some of the actual code details.