In [1]:

```python
import concurrent.futures

def quicksort(arr):
    """Standard quicksort algorithm with partitioning."""
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quicksort(left) + middle + quicksort(right)

def parallel_quicksort(arr, max_workers=4):
    """Parallelized quicksort using divide and conquer strategy."""
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]

    # Use concurrent.futures to sort left and right partitions in parallel
    with concurrent.futures.ThreadPoolExecutor(max_workers=max_workers) as executor:
        left_sorted = executor.submit(parallel_quicksort, left, max_workers)
        right_sorted = executor.submit(parallel_quicksort, right, max_workers)

        # Collect the sorted partitions
        left_result = left_sorted.result()
        right_result = right_sorted.result()

    return left_result + middle + right_result

# Example usage:
if __name__ == "__main__":
    arr = [3, 6, 8, 10, 1, 2, 1]
    sorted_arr = parallel_quicksort(arr)
    print("Sorted array:", sorted_arr)
```

```
Sorted array: [1, 1, 2, 3, 6, 8, 10]
```

In [ ]: