Exercise: Parallelize Prime Number Generation

You are given a simple JavaScript code for generating prime numbers within a given range. However, the current implementation is not optimized for performance. Your task is to refactor the code to use Node.js worker threads and parallelize the prime number generation process.

Instructions:

1. **Refactor the code**: Modify the existing code to use Node.js worker threads to parallelize the prime number generation process.
2. **Optimize performance**: Ensure that the prime number generation is split among multiple worker threads to take advantage of multi-core processors and make the overall process faster.
3. **Communication**: Implement a mechanism for the worker threads to communicate their results back to the main thread. Consider using the `workerData` and `parentPort` features provided by Node.js worker threads.
4. **Testing**: Verify that the refactored code produces the correct list of prime numbers for the given range.
5. **Sharing the Repository Link**: Share the link to your public repository. You can either send it via email or communicate it through your preferred channel for code sharing.

Base Code:

```
const min = 2;
const max = 1e7;

const primes = [];

function generatePrimes(start, range) {
    let isPrime = true;
    let end = start + range;

    for (let i = start; i < end; i++) {
        for (let j = min; j < Math.sqrt(end); j++) {
            if (i !== j && i % j === 0) {
                isPrime = false;
                break;
            }        }
        if (isPrime) {
            primes.push(i);
        }

        isPrime = true;
    }
}

generatePrimes(min, max);

const message = "Prime is : " + primes.join(" ");
console.log(message);
```