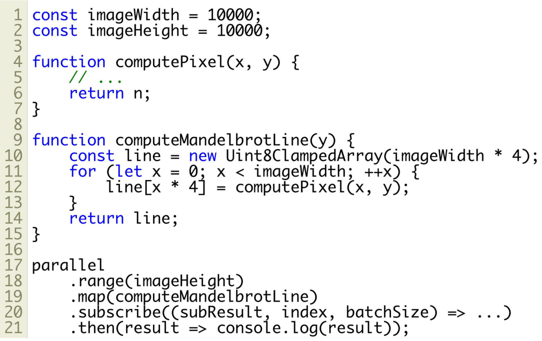
Micha Reiser

Students	Micha Reiser
Lecturers	Prof. Dr. Luc Bläser
Advisors	- -
Topic	Software and Systems

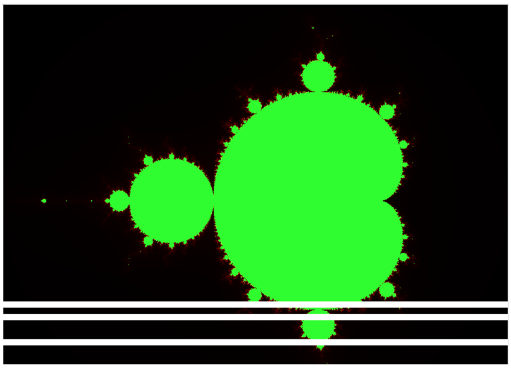
Parallelize JavaScript Computations with Ease

By abstracting the message-based programming model

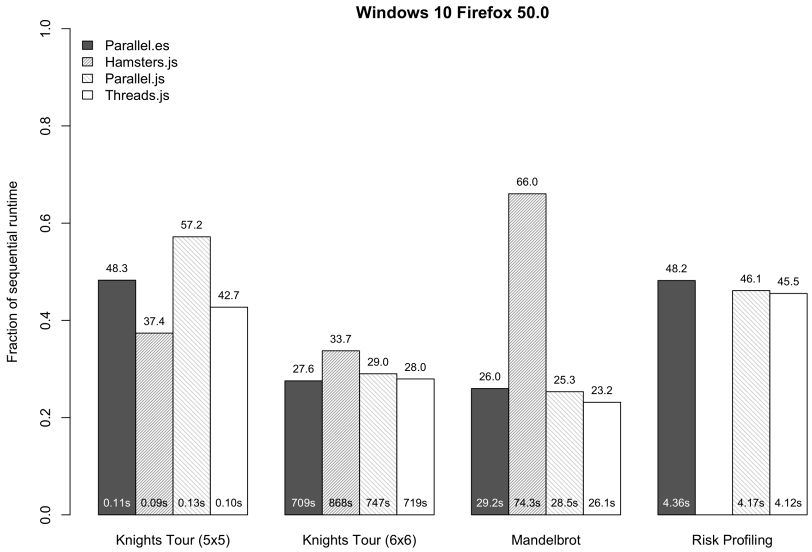


```
const imageWidth = 10000;
const imageHeight = 10000;

function computePixel(x, y) {
    // ...
    return n;
}

function computeMandelbrotLine(y) {
    const line = new Uint8ClampedArray(imageWidth * 4);
    for (let x = 0; x < imageWidth; ++x) {
        line[x * 4] = computePixel(x, y);
    }
    return line;
}

parallel
    .range(imageHeight)
    .map(computeMandelbrotLine)
    .subscribe((subResult, index, batchSize) => ...)
    .then(result => console.log(result));
```

Mandelbrot Implementation using Parallel.es



In Progress Rendering of the Mandelbrot



Benchmark Results showing that Parallel.es performs well

JavaScript evolves to a general-purpose language. Simultaneously, the complexity of its applications is rising, demanding for even more computational resources that can no longer be satisfied by a single-threaded runtime system. However, the JavaScript community has not widely employed multithreading because the available standards are platform-dependent and enforce a messaging-based programming model.

This paper presents Parallel.es, a platform-independent type-safe API and runtime system allowing to create multithreaded applications in JavaScript with ease. The runtime system abstracts the messaging-based programming model for a seamless integration into existing program code. Background tasks are defined by standard JavaScript functions and are executed concurrently in background threads. The runtime system further offers a reactive API simplifying the parallelization of data-stream-based operations by facilitating automated work partitioning and result joining.

The evaluation shows that the runtime system performs well compared to related work. Nevertheless, the proposed system is mainly superior because of its seamless integration into existing code and the type-safety of its API. Moreover, it offers the same debugging experience as for sequential code.