university of groningen

faculty of science and engineering

Highly Distributed In-Browser Computing

Bachelor's Thesis

July 2019

Authors
George Argirousis
Tolga Parlan

Primary Supervisor
Who?

Secondary Supervisor
Frank Blaauw

CONTENTS

ABSTRACT

Here be the abstract

1

INTRODUCTION

Modern web relies increasingly on Javascript, which is used almost ubiquitously in client-side of any website and is an increasingly popular tool for server applications through Node.js. Client-side Javascript often has the purpose of making websites more reactive to user actions by offloading some application logic to the client's browser. This approach is increasingly popular due to user computing devices getting ever more powerful [STATS], and serious innovations in Javascript engines [MENTION WHAT INNOVATIONS] and related technologies [REACT etc] making it ever more sensible to invest development time into shifting computational tasks to the client-side [EXPLAIN THE ADVANTAGES OF THIS MAYBE?]. This has been exposing a considerable amount of computing power to the website owners in the form of their users' devices, since presumably everyone visiting their page run code written by them.

[SUPER COMPUTER COMPARISONS OF THIS COMPUTING POWER FOR LARGER WEBSITES?]

Distributing a large pool of computing tasks to many computers across the internet and letting them run the calculations is an idea which is now more than two decades old and includes successful projects such as BOINC [REFERENCE, FIND OTHERS]. Such projects have been usually termed **Voluntary Computing** since the participants would actively volunteer to run such tasks on their machines, usually via signing up on a website and downloading a purpose-made program [BBHOFVC paper]. [TALK ABOUT BROWSER BASED VERSION OF THIS]. This model has a track-record of some serious achievements such as [@HOME projects].

Another idea in this field of distributing tasks to browsers, which comes from a similar premise while addressing a different situation is given the name **Gray Computing** [2]. Gray computing refers to a non-voluntary category of task distribution where users don't explicitly consent to running the given computations on their browsers, but the implicit consent which comes from the fact that any website can run arbitrary Javascript on it's visitors computers is used to create a similar architecture. This environment presents different technical challenges and opportunities, which are explored in the next chapter.

The lion's share of revenue for most websites that do not explicitly sell products come from advertisement [FIND STATS]. Ads can present a series of complications to internet users, [MAKE A LIST, FIND STG ABOUT THIS]. The negative user reaction against ads is tangible in ever growing use of ad-blockers [FIND STATS]. [RESEARCH IF ADS DECREASE USER ENGAGEMENT]

This project aims to develop a powerful, simple and highly customizable open-source gray computing tool which can help to un-

earth the aforementioned underused computing potential of billions of devices that access internet via web browsers. We envision that harnessing this processing power, with the right tools, can create a serious alternative revenue stream for many websites, nullifying or decreasing the need for ads, while serving useful scientific or non-scientific purposes.

For these ends, we have implemented an npm package [1] which integrates seamlessly with any Node.js http server and has an easy to use and highly customizable interface for programmers, behaving like a native Node.js package. We believe that this unopinionated approach that allows other implementers to run their own tasks in their own way and even change our project code to better suit themselves is a serious advantage over other similar projects, and can help to resolve the relative lack of real life popularity of past browser based distributed computing projects. We discuss this topic more in the next chapters.

Our code is open-source and can easily be reached by anyone who wants to contribute with new features or performance improvements, or simply fork from our project and use it as a starting point in their own projects. With this aim we are housing our project as a public repository on github and [TALK ABOUT LICENSE]

Going further, this paper will be structured as follows: In Chapter 2 we give a summary of related work, explaining how we have benefited from the design and technology choices of similar projects to ours. The chapter further quotes from survey papers which have examined many past projects, and goes on to explain in which ways we have attempted to differ in our project in order to produce a solution that can be adopted easily by people for diverse tasks. In Chapter 3 we summarise our code architecture, and talk about the main components of our program in more detail. Chapter 4 focuses on evaluating different types of tasks and their performance on our system. Finally in Chapter 5 we present our conclusions and in Chapter 6 we present ideas that can be considered to improve the project in the future work.

2

RELATED WORK

3

ARCHITECTURE

4

EVALUATION

5

CONCLUSION

6

FUTURE WORK

BIBLIOGRAPHY

[1] npm.

[2] Biswajit Pathak and Debajyoti Barooah. Texture analysis based on the gray-level co-occurrence matrix considering possible orientations. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2(9):4206–4212, 2013.