1 Motivation

Over the past thirty years, the ubiquity of the computer has spurred every scientific and technological field to become increasingly data-driven. As processors have become more powerful, the size of the datasets they are expected to deal with has also correspondingly increased. Because of this, computing power has become a major bottleneck for researchers in fields from bioinformatics to cosmology. Projects like Folding@Home and BOINC@Home have sought to address this problem by enlisting volunteers to install software on their personal computers that donates idle cycles to solving highly parallel research problems. Although these projects have tremendous potential, they have suffered from lack of adoption. Cusack et al. analyze the reasons for this unpopularity, and find that chief among them are "lack of awareness, limited demographic, and lack of technical savvy." Knowledge of these projects is largely limited to technically savvy users (many of them involved in the research community themselves), and others who might want to join find the difficulty of installing complex software a high barrier to entry.

We suggest that a Javascript-based distributed computing platform that executes code in users’ browsers solves for many of the adoption problems that have plagued such platforms. 85% of Americans today use the Internet, and visiting a website represents a far lower barrier to entry than installing and configuring software. Moreover, a Javascript-based platform has the potential to be embedded in other websites, allowing
website owners to donate the idle cycles of their users to solving these problems.

2 Background

Until recently, such a platform would be incredibly impractical. Code written in Javascript has traditionally run at just a few percent of the speed of equivalent code in C or C++ compiled for the native platform. However, the development of asm.js, a compilable subset of Javascript supported by modern browsers has brought Javascript execution speed to approximately 65-70% of the speed of native code.

Unlike normal Javascript, asm.js is not intended to be written by hand. Instead, compilers are used to take code written in statically typed languages and compile them to asm.js. One of these compilers is Emscripten, a LLVM backend developed by Mozilla. Because Emscripten can compile code written in C or C++, as a substantial amount of high performance computing code is, it is now possible for this high performance code to be executed in the browser at reasonable speeds, without rewriting any source. However, Emscripten has no support for MPI or any robust form of parallelism, presenting a major barrier to adoption for parallel computing applications.

3 Proposal and Deliverables

We propose extending the Emscripten compiler to support elements of the OpenMP standard. By interfacing with Atlas, the distributed computing platform developed by Jason Brooks last semester, we will distribute parallelizable jobs to a grid computing network of browsers and return the results to the executing process. Deliverables will be as follows:

1. Source code of modifications to the Emscripten compiler. First, the OpenMP "parallel" and "parallel for" pragmas will be supported for the read-only shared
memory case. After this is complete, message passing will be implemented, along with OpenMP synchronization and sharing clauses to support the full OpenMP standard. If time permits, the use of memory usage verification techniques will be explored in order to minimize the number of memory accesses over the network. Additionally, we may explore the use of verifiable computing techniques in order to prevent data tampering.

2. Working server allowing the upload and execution of distributable code on a test grid network of browsers. This will be completed in collaboration with Jason Brooks, who is continuing work on Atlas this semester.

3. Research paper summarizing design and providing performance benchmarks of the system and status quo alternatives.

References
