CIVIC (Central Intelligence Virtualization Instruction Cluster)

A capstone project report submitted in partial fulfillment of the requirements for the degree of

Masters of Engineering

In the Department of Computer Science Graduate Program,

College of Engineering & Applied Science

04/11/2025

Margeson, Jack

1 Abstract

Distributed computing is a design methodology that enables digital storage and data processing to be completed across multiple devices, often on separate networks. CIVIC (Central Intelligence Virtualization Instruction Cluster) allows organizations (labs, campuses, businesses, etc.) to deploy their own instance of a cooperative computational system. CIVIC's platform streamlines program development for organizations by leveraging virtualization technologies, and the user-friendly and flexible workflow allows participants to easily volunteer computational resources to CIVIC instances.

The overall goal of this project is to address the limitations of current volunteer distributed computing programs and open new possibilities for research and development. Unlike existing frameworks that require specific programming languages or hardware architectures when writing applications, CIVIC provides a flexible and language/architecture-independent design.

Researchers can deploy their existing code to a CIVIC instance with minimal changes, while participants can easily contribute computational resources. This is achieved through the creation of models, a set of instructions that can be used by client machines to create virtualizations. These virtualizations, referred to as citizens, are worker containers capable of receiving fractions of input data and executing code to generate output. Each fraction of data generated by the server is called a duty, and each duty can be assigned to one or more citizens depending on the result verification needs of the project.

The CIVIC Server manages model distribution and communication with citizens, as well as processing, validating, and assimilating incoming duty results–ensuring efficient and scalable distributed computing by minimizing required human interaction from both organization and participant perspectives.

2 Introduction

In the field of distributed computing, one subset of applications is volunteer computing. Services that fall under this category, including SETI@home, Folding@home, and BOINC, allow users to donate their idle computing resources to scientific research projects. These projects typically require large amounts of computational power to process data, and volunteer computing allows researchers to leverage the collective power of many individual computers. For example, one of the most popular volunteer computing applications, Folding@home, is used

to simulate protein folding to contribute to scientific research in the field of biology through the study of diseases such as COVID-19 and Alzheimer's disease. The protein folding process is a complicated mathematical simulation that requires significant computational resources to run. For problems like these, volunteer computing is a powerful methodology to leverage the computing power of several machines to independently work on fractions of the problem. Utilizing volunteer computing in a scenario like this reduces the overall time that it would take one machine to complete the task.

However, there are limitations to some of the existing volunteer computing frameworks. For example, we can examine BOINC, the the Berkley Open Infrastructure for Network Computing. BOINC is a framework that allows researchers to create their own volunteer computing projects and allows volunteers to donate computational resources to any of these projects by downloading the client and selecting from a list of available causes. One of the big issues when it comes to creating a research project on BOINC is the fact that programs that are to be distributed to volunteers must be packaged in a very specific way. The BOINC framework provides three options for application developers:

- Native applications, in which the program must be written in a language that supports the official BOINC API and compiled for a target architecture.
- Wrapper applications, in which a program is written in a language that does not support the official BOINC API but can be run as a sub-process of a wrapper application that handles input and output.
- Virtual applications, in which a program is run in a virtual machine that is managed by the BOINC framework through external scripts and programs such as VirtualBox.

Each of these options have their own limitations. With native applications, the developer of the program must write in a programming language that either officially supports (C/C++) or unofficially supports (FORTRAN, Python) the BOINC API interface. For both native and wrapper applications, the program that is written must be compiled for a target architecture before distributed to a client. The implication of this is that if a researcher would like to allow participants to contribute to the project, they must first compile any programs written for the project for each architecture that they would like to support, i.e. x86 Windows, x86 Linux, ARM Linux, Android, etc. This can be a time consuming process, especially when program changes

are made and all of the architectures must be recompiled. Additionally, code changes must be made to programs when dealing with certain functionality, such as differences in how system commands are handled between Windows and Unix based operating systems. The third option, virtual applications, mitigates some of the problems with native and wrapper applications. These types of BOINC applications are able to execute in a virtual machine, which allows the program to be written in any language and run on any architecture. However, this option entails significant additional setup and configuration with an external program, VirtualBox, which can introduce additional complexity, runtime overhead, and download size for volunteers.

These limitations are not exclusive to BOINC—several other volunteer computing frameworks have similar restrictions. In this capstone report, a new alternative to existing volunteer computing frameworks is proposed. CIVIC (Central Intelligence Virtualization Instruction Cluster) is a framework that utilizes virtualization technology to allow researchers to easily create and deploy volunteer computing projects. By utilizing virtualization, CIVIC allows for the development of language and architecture independent applications. In combination with a binary application, researchers can use the framework to parse and create datasets to create models for distribution. Volunteers looking to contribute to research are able to donate their computational resources to a project by creating a ”citizen”—a virtual client program that runs in it's own isolated container environment. These citizens automatically download and execute model instructions given from the server with a subset of the input dataset and return the results to the server. The primary goal of CIVIC is to provide a framework that allows easy configuration and deployment of volunteer computing projects for researchers, while also providing a simple and user-friendly experience for volunteers looking to contribute to research. The framework is designed to be flexible and extensible, allowing researchers to easily adapt it to their specific needs.

3 Methods

To achieve the goals of this project, several technologies were utilized in the creation of CIVIC. The first of these technologies is Docker, a containerization technology that allows for programs to be run in an isolated environment. The Docker Engine