Towards Building Computing Clusters of Web Browsers

Yao Li[1] Yasushi Shinjo[1]

1. Introduction

It is widely practiced to connect PCs over a LAN to form small HPC clusters. Conventional cluster management software such as Apache Mesos and OpenHPC requires a dedicated OS installation, making the setup process complex. While dual-booting allows regular PCs to serve as cluster nodes, this prevents their use for normal tasks like web browsing.

Volunteer computing projects like SETI@home and Folding@home have shown the potential of utilizing idle computational resources. However, these systems typically require specific software installation and face cross-platform compatibility challenges.

We propose a web browser-based approach to building computing clusters with the following goals:

- Developing a scalable architecture for dynamic resource pooling without complex infrastructure, targeting both small HPC clusters and volunteer computing environments
- Ensuring zero-installation deployment through standard web browsers
- Enabling efficient utilization of idle computational resources across the network
- Providing secure execution through browser-based isolation

2. Proposed Method

Figure 1 illustrates the system architecture of our browser-based computing cluster. The system consists of three main components:

- **Management Web Server**: A central server that coordinates the cluster operations, including task distribution, resource monitoring, and node management. The server communicates with nodes through WebRTC.
- **PC Nodes**: Standard computers connected to the network, each running a web browser.
- **Browser Applications**: Each browser runs our application that provides an isolated runtime environment for executing distributed tasks.

Our system architecture is built on the following key technologies to ensure efficiency, security, and scalability:

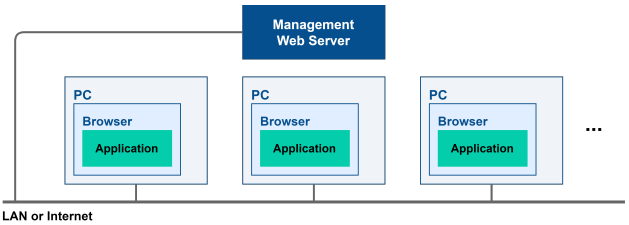

Fig. 1 Cluster of Web Browsers.

- WebAssembly provides high-performance task execution while maintaining platform compatibility
- WebRTC enables direct peer-to-peer communication between nodes for efficient data transfer
- Dynamic resource management ensures optimal distribution of computational loads across the cluster

We designed the system to be scalable, allowing for seamless joining and leaving of nodes to the cluster. This architecture provides a foundation for building efficient and accessible browser-based computing clusters that can leverage idle computational resources across different platforms and environments.

3. Implementation

Our implementation consists of three main components as shown in Figure 2: the management web server, the browser-based virtual machine, and the communication module.

3.1 Management Web Server

The management web server implements two core modules:

- **Task Scheduler**: Handles task distribution and load balancing across nodes. It fragments computational tasks based on node capabilities and manages task allocation.
- **Resource Monitor**: Maintains performance profiles and monitors node status continuously. It tracks CPU usage, memory availability, and network conditions to optimize resource utilization.

3.2 Browser-based Virtual Machine

We support the following two types of execution environments of applications:

[1] University of Tsukuba

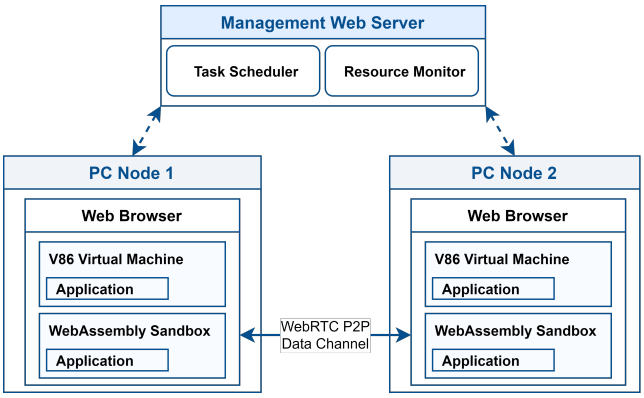

Fig. 2 System Components.

- **Using V86-Based Virtual Machine**[*1]: Provides x86 emulation capabilities directly in the browser. This enables task execution in a manner similar to conventional cluster management software.
- **Running a single WebAssembly binary in the WebAssembly Sandbox**: Ensures secure and efficient execution through memory isolation and secure execution boundaries and cross-platform compatibility across different operating systems.

3.3 Network Communication

The system implements two communication channels with distinct responsibilities:

- **RPC**: Handles control communication between the management server and nodes, including:
  - Task distribution and status updates
  - Resource monitoring and node management
  - Node registration and heartbeat signals
- **WebRTC P2P**: Enables direct peer-to-peer communication between nodes through:
  - Data channels for efficient resource sharing
  - Direct node-to-node task coordination
  - Low-latency data transfer between peer nodes
  - NAT traversal for direct connectivity

This dual-channel approach optimizes communication efficiency by:

- Using RPC for reliable server-controlled operations
- Leveraging WebRTC for direct peer communication to reduce server load
- Maintaining centralized control while enabling decentralized data exchange

3.4 Current Progress

Our initial implementation has established core functionalities including:

- Basic virtual machine operations through V86
- WebRTC peer discovery and connection establishment
- Task distribution through the management server
- Resource monitoring and basic load balancing

Ongoing work focuses on enhancing system stability, improving task scheduling efficiency, and implementing more sophisticated resource management strategies.

[*1] http://copy.sh/v86/

4. Related Work

BOINC established a framework for volunteer computing projects [1]. Despite its robust infrastructure, BOINC requires client software installation. Our browser-based solution removes this barrier by running directly in web browsers.

Golem provides a decentralized computing power marketplace for renting computational resources [2]. While effective for dedicated computing tasks, our approach simplifies participation by eliminating the need for specialized software installation.

Apache Mesos [3] demonstrates effective resource management. Building upon these concepts, our system combines WebAssembly's efficiency with browser accessibility to create a lightweight, platform-independent computing environment.

5. Conclusion

In this paper, we proposed building web browser-based computing clusters that provide zero-installation deployment through standard web browsers, cross-platform compatibility, and secure execution through WebAssembly sandboxing.

Our initial implementation has established core functionalities including WebRTC peer discovery and basic virtual machine operations. While challenges remain in areas such as browser performance heterogeneity and connection stability, this approach demonstrates the feasibility of leveraging web browsers for distributed computing applications.

References

[1] Anderson, D. P.: Boinc: A system for public-resource computing and storage, *Fifth IEEE/ACM international workshop on grid computing*, pp. 4–10 (2004).
[2] Golem Network: Official Website, `https://www.golem.network/`. Accessed: 2024-11-04.
[3] Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A. D., Katz, R., Shenker, S. and Stoica, I.: Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center, *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, pp. 1–14 (2011).