# RAI: A Scalable GPU Submission System for Machine Learning Applications

Abdul Dakkak[1], Carl Pearson[2], Cheng Li[1], Wen-mei Hwu[2]
Department of Computer Science[1],
Department of Electrical and Computer Engineering[2],
University of Illinois at Urbana-Champaign

## Introduction

The success of GPU-enabled machine learning applications has led the GPU programming classes at the University of Illinois to incorporate open-ended machine-learning projects. Students may

- Modify compiler invocations
- Depend on arbitrary software
- Use profiling tools
- Write tools for managing data
- Use debugging tools
- Have control over execution time

RAI is an open-source GPU programming system that realizes these objectives to be met in a cost-effective, scalable, and secure way.

## Features & Benefits

### Security

To isolate student environments, RAIcreates unique, virtualized, transient Docker containers for each submission.

### Configurability

To provide as much flexibility as full system access, RAI accepts bash commands, and non-interactive applications may be executed within user defined containers.

### Accessibility

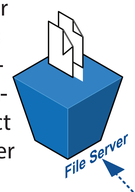
Since remote students do not have access to physical computing facilities, GPUs, or esoteric programming environments, RAI provides Windows, Linux, or macOS clients to develop machine learning applications over the internet.

### Scalability and Cost

Large classes and deadlines create bursty system loads, RAI scales-out worker nodes only as needed to reduce costs. It can also be configured to utilize local clusters with expasion to cloud resources during bursts.

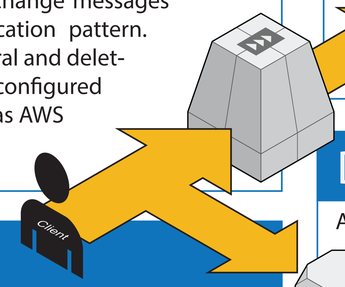
## File Storage Server

When a student submits a job to RAI, the project directory gets uploaded to a file server. Upon job completion, the worker node's output directory is also uploaded to the file server and is available to download. This way students have access to any output or logging files the submission generates. The file server is also used by instructors to download all files tagged as final project submissions. RAI supports any file storage server which is AWS S3 compatible.




## Message Broker

The message broker arbitrates communication between clients and workers to maintain fairness and resiliency for submissions. Workers and clients connect and subscribe to a distributed queue system on the broker. Both the RAI clients and workers subscribe to the message broker and exchange messages using a publish/subscribe communication pattern. Messages and queues can be ephemeral and deleted once a timeout occurs. RAI can be configured to use broker and queue servers such as AWS SQS, AWS SNS, RabbitMQ, and NSQ.



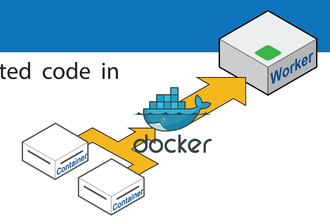

## Client

The RAI client is an executable downloaded by the students and runs on the students' machines. The executable requires no library dependencies and works on all the main operating systems and CPU architectures. Both features reduce the likelihood that students will have technical difficulties running the client. Students use the RAI client to interact with the system and to submit jobs.

RAI clients require authentication keys that uniquely identify the user to run. Authentication keys are generated by teaching staff through a companion utility.

## Worker Node

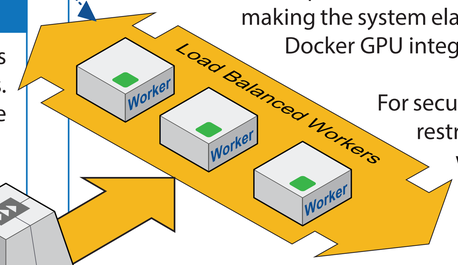
The RAI worker acts as an agent that runs submitted code in Docker containers. To maximize configurability, the submitted jobs can specify arbitrary Dockerfiles as the container. The worker can be used with and without the nvidia-docker volume plugin.




## Load Balanced Worker Nodes

The RAI worker acts as an agent that starts a sandboxed environment to execute students' code. Multiple worker nodes can exist within the RAI environment, thus making the system elastic and scalable. Because of current limitations of Docker GPU integration, the worker must run on a Linux system.

For security and isolation, Docker containers may be given restricted access to CPU, memory, filesystem, and networking resources. The sysadmin may also choose which Docker images are allowed to be used on the worker.




## Database

A database, stores meta-information such as execution times, submissions, runtimes, and logs. The information in this database is useful for grading or any other coursework auditing process. The database is also used to store team ranking if the class project is a competition. RAI supports relational and NoSQL databases such as MySQL, PostgreSQL, MongoDB, and AWS Dynamo.


## Conclusion

RAI is a scalable job submission system designed for machine-learning and other CPU and GPU workloads. RAI's design addresses challenges of scalability, configurability, security, and cost in delivering a flexible parallel programming environment. RAI has been used to offer two machine-learning projects and a wide variety of other submissions in GPU programming courses at the University of Illinois Urbana-Champaign.