Toward High-Performance Simulation of 1000 Drones

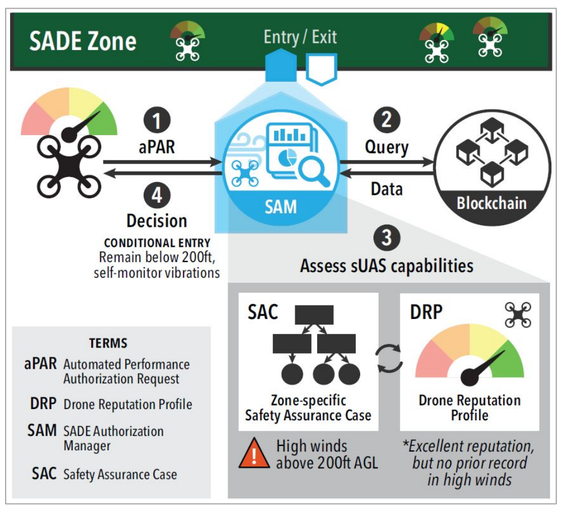
Md Saiful Islam and Douglas Thain
Cooperative Computing Lab, University of Notre Dame
{mislam5, dthain}@nd.edu

Abstract

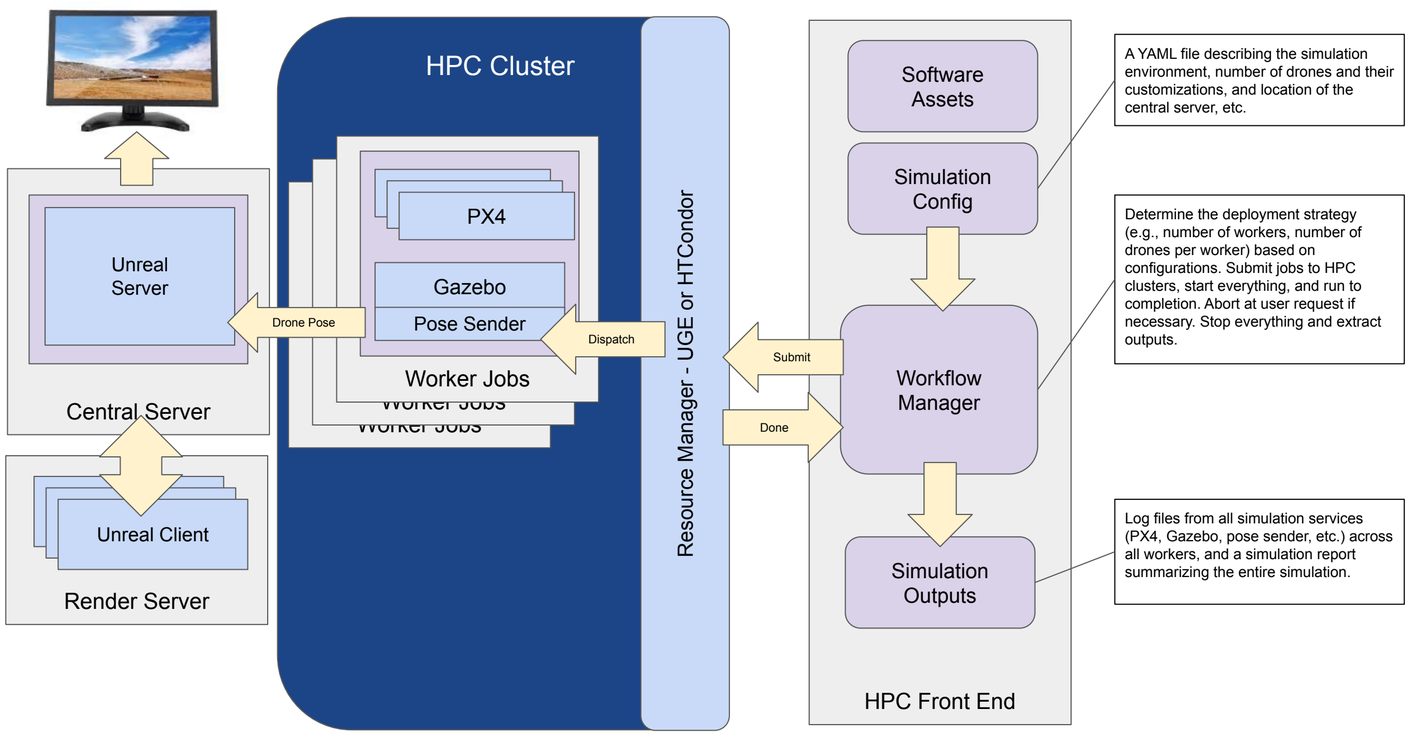
Traditional drone simulation tools often face challenges in concurrent execution due to their reliance on predefined ports and filesystem resources. This can lead to conflicts when multiple simulations run concurrently on the same node in HPC. Furthermore, HPC security policies typically prevent regular users from creating isolated network namespaces, hindering the ability to simulate large-scale drone scenarios on HPC systems. We have designed a system that overcomes these limitations and facilitates large-scale distributed simulations on HPC infrastructure. Our solution reserves required ports, dynamically resolves resource conflicts, and provides a clear workflow for dependent tasks between and within simulations. This work is a key part of the simulation component of SADE, a system designed to safely manage drone traffic, automating authorization decisions for drone flights, offering speed, transparency, and scalability for a large number of drones.

SADE Project

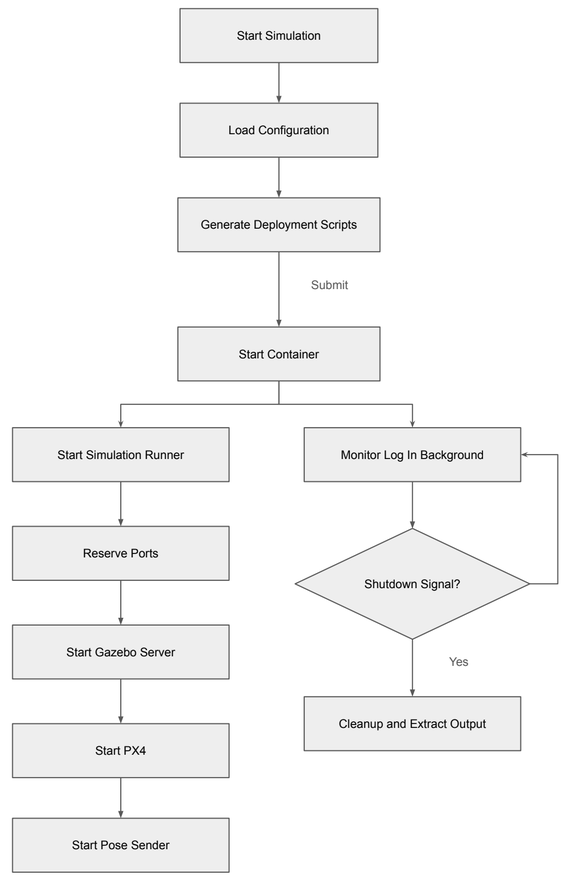
The SADE project aims to ensure safe and privacy-respecting flights for small Unmanned Aerial Systems (sUAS) in managed airspaces through automated and fair decision-making. It features onboard analytics for capturing flight behavior with privacy-aware reputation models, reliable communication via chirper, and fair, transparent decision processes based on safety cases. Additionally, the project includes a scalable simulation environment to enhance the accessibility and effectiveness of SADE Zones.



Architecture



Workflow



Challenges

Concurrent Execution: Simulation applications lack support for concurrent runs, leading to execution conflicts.

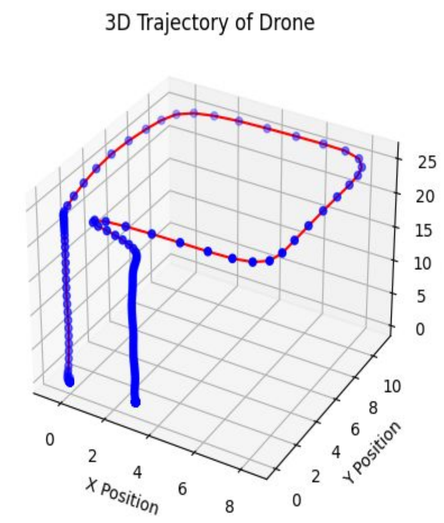
Policy Restrictions: HPC rules prevent creating isolated network spaces, limiting large-scale simulations.

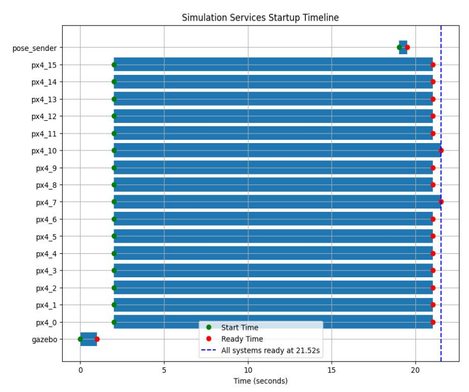
Port Management: Ports must be reserved ahead of simulation to prevent conflicts.

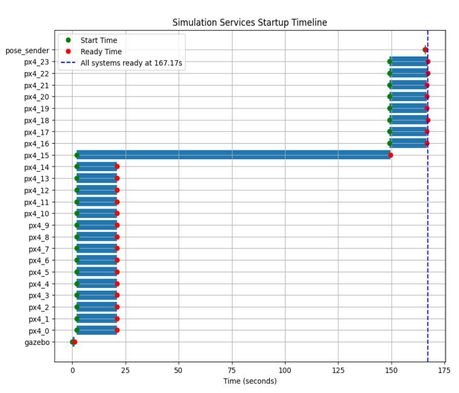
Synchronization: Coordination is required at startup due to interdependencies and delays.

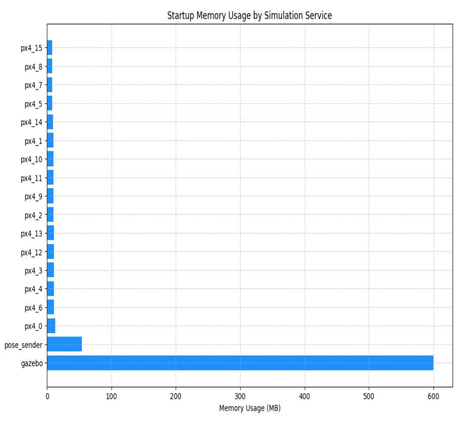
Worker Allocation: Optimizing the number of simulation workers based on simulation scale is crucial for efficiency.

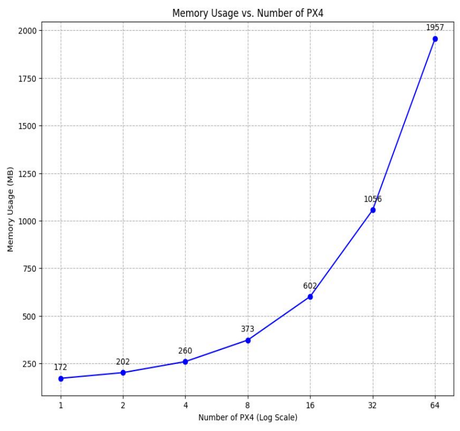
Initial Results

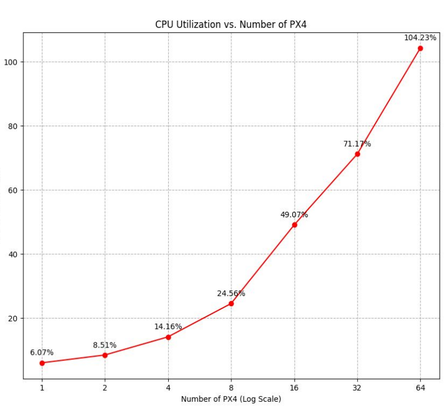












Acknowledgement: This work was supported by NASA grant 80NSSC23M0058, "A Safety-Aware Ecosystem of Interconnected and Reputable sUAS"