Distributed Treasure Hunt

Team Members:

1) Ajita Shrivastava
2) Andy Kuei
3) Sree Hari Karri

Description:

Everyone has played a multiplayer online game at some point or the other. They may look simple but the engineering that goes into handling consistent states among all users is no joke. Games usually suffer from scalability issues, server lag, and state inconsistencies. We aim to build a project that can achieve a consistent and responsive user experience while also being a competitive game that users can engage in. Our objective is to address these challenges and support numerous concurrent players while maintaining a consistent experience.

Motivation:

The project is intriguing due to its complex requirements for real-time user interaction in a distributed system. It's challenging because it demands efficient synchronization, fault tolerance, and immediate reflection of shared state changes.

Previous Work:

To support efficient synchronization and fault tolerance without the use of a centralized server, we could utilize serverless computing like the system provided in [1]. Most notably, the authors noted that serverless computing could be used to generate game states or speculative game states to increase user quality of life and decrease latency. This also had the byproduct of supporting more users than previously possible(150 as opposed to 10) and was tested using the game Opencraft(Minecraft). This is also supported by [4] where mobile games are computed on cloud servers to reduce the computations of a single client. In fact, Kubernetes is shown to support the running of game servers with no technical limitations and would be a feasible resource to support a server[6]. Multiple clients may “collide” by accessing the same resource or room. We would address this using the idea of shared servers or spaces proposed by [3] which would predict when two clients get close together and when a collision might occur. Other fault tolerance measures are provided in [1], where the server’s game state takes priority over user actions in the event of user latency or user crashes.

Initial Ideas:

The idea is to have a game that asks the user riddles based on cars. It will start off with simple riddles which point the user to a webpage, inside the webpage the user should use a torch light effect in a dark garage to find the correct car part, which is the treasure. This treasure is limited in number, so once the treasure is all found, the round automatically ends and the users are taken back to the lobby. If a client dies in between the game, he should be ejected from the lobby and when he joins again, his old state should be restored.

Server Responsibilities:

- Manage user registration and ensure unique usernames.
- Maintain game state (riddles, timer, progress).
- Control **mutual exclusion** for treasure claims.
- **Replicate** game state for high availability.
- Scale server instances based on client load.
- Implement **gossip protocol** for state updates.

Client Responsibilities:

- Render game interface and process user inputs.
- Participate in gossip protocol for receiving and forwarding updates.
- Handle **concurrency** for treasure claims.
- Subscribe to updates on game state changes.

Milestones:

March 7 - 14: Division of components, research, initial setup
March 15 - 21: Setup user login and homepage and server, aim to establish communication between both.
March 22 - 30: Implement lobby system and game design
April 1 - 5: Spring Break
April 6 - 13: Test gossip protocol, mutual exclusion.
April 14 - 21: Deploy on k8s and test scaling.
April 21 - 31: Report + wrap up.

Resources:

SW: Next.js, Node.js, Socket.IO, MongoDB, Docker, K8s, GCP/AWS
HW: Laptops

Citations:

1. J. Donkervliet, J. Ron, J. Li, T. Iancu, C. L. Abad and A. Iosup, "Servo: Increasing the Scalability of Modifiable Virtual Environments Using Serverless Computing," 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS), Hong Kong, Hong Kong, 2023, pp. 829-840, doi: 10.1109/ICDCS57875.2023.00075.
2. K. Luo, T. Ouyang, Z. Zhou and X. Chen, "Behavior Tree-based Workflow Modeling and Scheduling for Serverless Edge Computing," 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS), Hong Kong, Hong Kong, 2023, pp. 955-956, doi: 10.1109/ICDCS57875.2023.00100.
3. M. Assiotis and V. Tzanov, “A Distributed Architecture for Massive Multiplayer Online Role-Playing Games,” 2005. Available: https://pdos.csail.mit.edu/archive/6.824-2005/reports/assiotis.pdf
4. M. Zamith *et al.*, “A Distributed Architecture for Mobile Digital Games Based on Cloud Computing,” Nov. 2011, doi: https://doi.org/10.1109/sbgames.2011.13.
5. J. Lundgren, ‘Kubernetes for Game Development : Evaluation of the Container-Orchestration Software’, Dissertation, 2021.
6. K. Aning and K. L. Mannock, "An architecture and implementation of the actor model of concurrency," 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA), Larnaca, Cyprus, 2017, pp. 1-6, doi: 10.1109/IISA.2017.8316391.