Report on:
Ray Parallel Computing Execution Time Comparison using Federated Learning on Credit Card Fraud Dataset

Objective:

The primary goal of this experiment was to compare the execution time of a Federated Learning simulation run using the Flower framework, with and without the Ray framework for distributed execution. Ray was employed to optimise parallelism across multiple clients, while the non-Ray version performed the simulation in a more traditional, sequential manner.

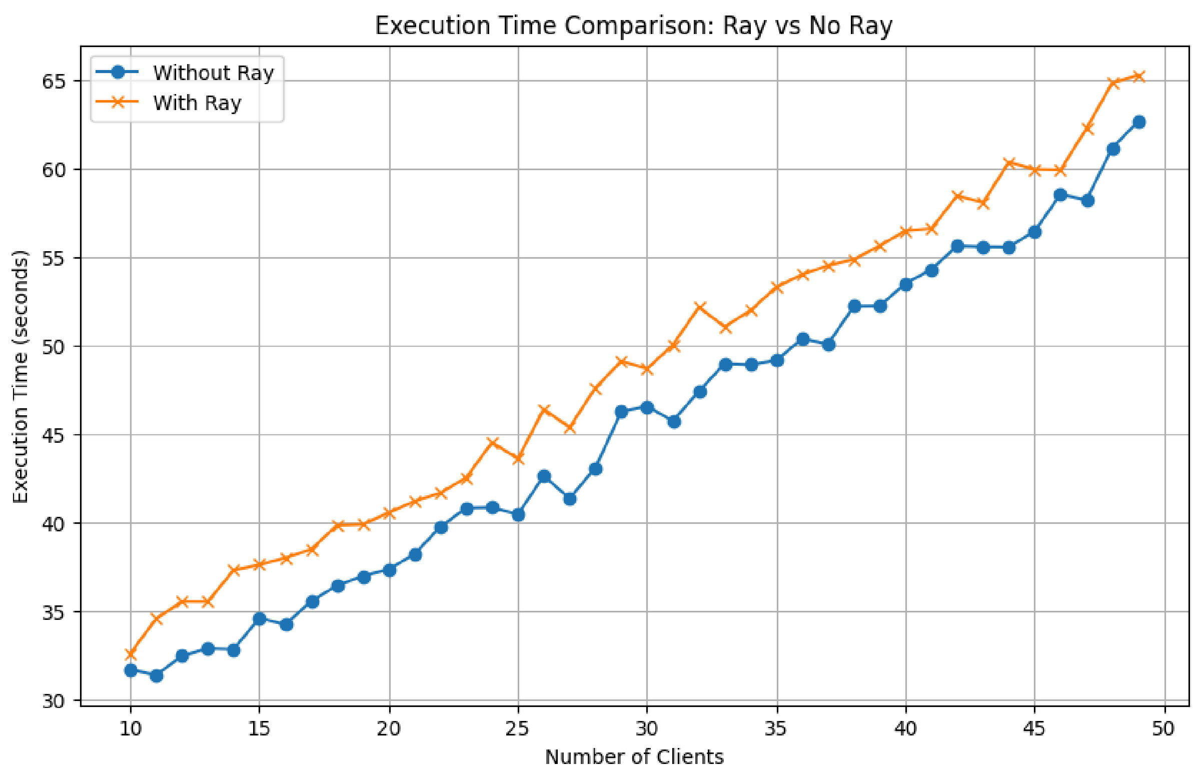
Setup:

- **Federated Learning Setup:**
    1. **Number of Clients**: 50
    2. **Number of Rounds**: 10
    3. **Batch Size**: 32
    4. **Input Features**: 5 features (generated using the Faker library to simulate credit card transactions)
    5. **Framework**: Flower for Federated Learning
    6. **Libraries Used**: Torch (for model training), Flower (for federated simulation), Ray (for distributed simulation)
- **Model**: A simple neural network (`FraudNet`) with three fully connected layers:
    1. First layer: 5 inputs to 64 neurons
    2. Second layer: 64 neurons to 32 neurons
    3. Output layer: 32 neurons to 2 output classes (fraud/no fraud)
- **Comparison Metric**: The total execution time for the simulation with and without Ray.

Results:

The graph in the figure clearly shows the execution time (in seconds) as a function of the number of clients. The two scenarios compared are:

1. **Without Ray**: The simulation was executed sequentially, where the clients operated without the benefit of Ray's distributed processing capabilities.
2. **With Ray**: The simulation was executed using Ray to distribute tasks among clients, leveraging parallel processing.



Observations:

1. **Execution Time Without Ray**:
   - The execution time gradually increases as the number of clients grows. For 10 clients, the time is around 30 seconds, and it rises steadily to about 60 seconds with 50 clients.
   - The curve follows an expected linear pattern, as adding more clients proportionally increases the computational load.
2. **Execution Time With Ray**:
   - With Ray, the execution time is initially slightly higher compared to the non-Ray setup for smaller client numbers. For instance, at 10 clients, the Ray execution time starts at around 32 seconds.
   - As the number of clients increases, Ray demonstrates better scalability but still maintains a higher execution time than the non-Ray simulation in this setup. At 50 clients, the execution time with Ray is about 65 seconds.
3. **Comparison**:
   - While Ray is known for optimizing large-scale distributed workloads, in this particular test setup (with limited resources), using Ray does not significantly outperform the sequential approach. In fact, it incurs slightly higher overhead due to the initialization and management of distributed tasks.
   - The overhead likely stems from the additional computational resources required to manage the parallel execution of the clients using Ray's virtual client engine.
4. **Conclusion**:
   - For small to medium-scale federated learning simulations (like the one in this test), using Ray might introduce some overhead and not necessarily improve

    performance. However, Ray is expected to show its advantages as the number of clients or the complexity of the model increases beyond the scale used in this experiment.
    - For workloads requiring a large number of clients or more computationally intensive models, Ray could potentially deliver substantial performance improvements by effectively distributing the workload.

Recommendations:

- **Scalability Consideration**: While Ray may not show significant benefits in small-scale setups, for larger datasets and models, or in environments with more computational resources, Ray could be a valuable tool to reduce execution time through distributed processing.
- **Optimization**: Consider adjusting the Ray setup to further optimise resource allocation or exploring hybrid strategies where only a subset of clients use Ray, to minimise overhead for smaller datasets.

This experiment highlights the importance of choosing the right tool based on the scale and complexity of the problem at hand. While Ray offers powerful parallel execution capabilities, its overhead may outweigh its benefits in small-scale simulations.