Distributed Chess Engine Proposal

Course: Distributed Systems

Team Size: 3

Programming Language: Python

Team Members:

- **2024201019** Shubham Raut
- **2024201051** Aniket Ransing
- **2024201070** Saurav Deshmukh

Introduction

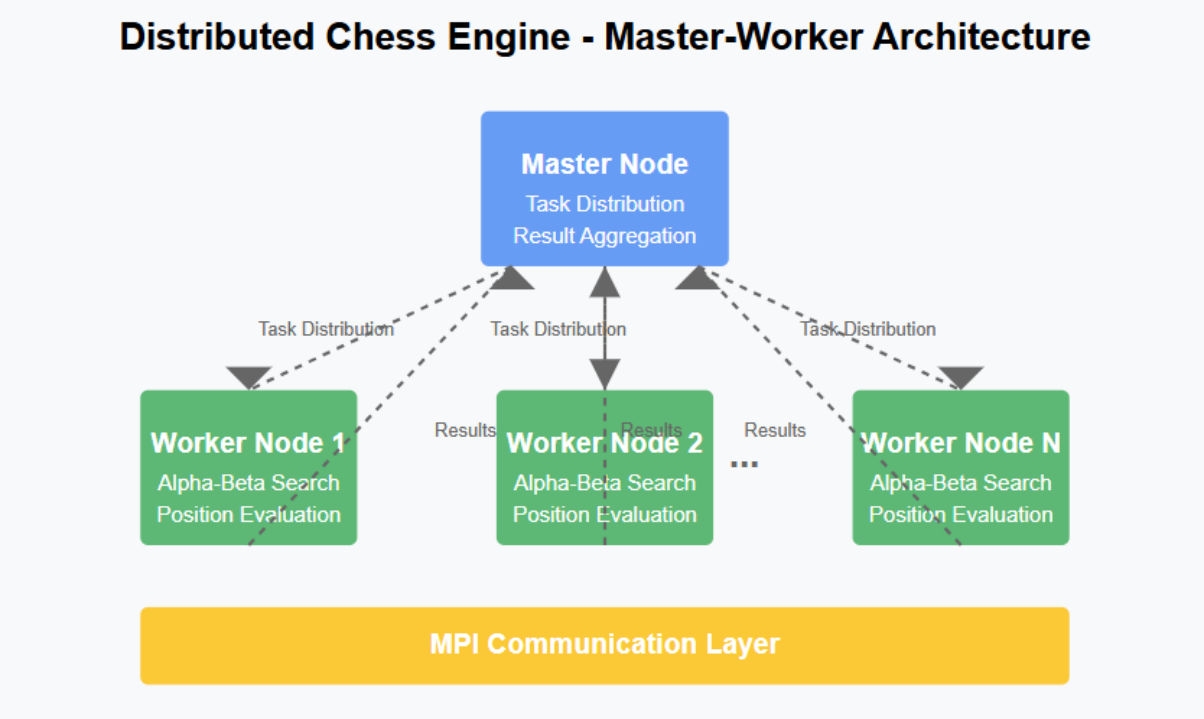
Chess engines are computational systems designed to analyze chess positions and generate optimal moves. The challenge in building an efficient chess engine lies in the massive search space and the need for real-time decision-making. A distributed chess engine leverages parallel computing to improve search efficiency and reduce computation time. By distributing the game tree exploration among multiple processors, we can significantly enhance performance and scalability.

Problem Statement

The primary challenge is to develop a distributed chess engine that efficiently explores the game tree using parallel computing and search optimization techniques. The engine should effectively distribute tasks, manage concurrency, and maintain workload balance to maximize computational efficiency. Additionally, the system must be robust to node failures and support dynamic load balancing.

Proposed Approach

We propose to implement a distributed chess engine using the **Master-Worker model** with **MPI (Message Passing Interface)** in Python. The master node will distribute tasks (game positions and search depth) to worker nodes, which will perform evaluations using algorithms such as **Minimax with Alpha-Beta pruning**. Workers will return their evaluation scores to the master, which will aggregate the results to determine the optimal move.



Future Considerations

While the initial implementation will use a **Master-Worker** architecture with **MPI**, we are exploring the possibility of enhancing the system with a **Peer-to-Peer (P2P) model** combined with **gRPC for inter-node communication**. This approach would increase fault tolerance and scalability by eliminating the single point of failure inherent in the Master-Worker model.

Tech Stack

- **Programming Language:** Python
- **Framework:** MPI (using mpi4py library)
- **Future Enhancements:** gRPC for P2P communication

Functionalities and Features

1. **Parallel Game Tree Exploration:** Efficiently search through possible moves using parallel processing.
2. **Dynamic Load Balancing:** Redistribute tasks among nodes based on their workload to ensure efficient utilization.
3. **Alpha-Beta Pruning:** Prune unnecessary branches in the game tree to reduce computation.

4. **Fault Tolerance:** Detect and handle node failures gracefully, redistributing tasks when needed.
5. **Move Evaluation and Ranking:** Evaluate board states and rank moves based on calculated scores.
6. **Communication and Coordination:** Seamless inter-node communication using MPI for data sharing and coordination.
7. **Performance Monitoring:** Track and analyze computational performance metrics and load distribution.
8. **Modular Design:** Easily integrate alternative search algorithms and communication protocols (like gRPC).

Deliverables

1. A fully functional distributed chess engine with parallel search capabilities.
2. Benchmark results comparing different configurations and models.
3. Documentation detailing system architecture, implementation, and performance analysis.
4. Future improvements including the transition to a P2P model with gRPC.

Conclusion

Our project aims to build a robust and efficient distributed chess engine leveraging parallel computing techniques. With a strong foundation using MPI and the flexibility to incorporate gRPC and P2P models in the future, this project will demonstrate the potential of distributed computation in game AI.