Algoritmo

- **Quicksort Paralelo**: Este algoritmo usa uma abordagem de dividir para conquistar, que é naturalmente apta para paralelização. Utiliza pivôs para o particionamento recursivo dos dados entre os processadores, não necessitando de sincronização durante a separação, pois cada sublista é processada independentemente.
- **Ordenação por Amostragem**: Divide o arquivo de entrada em múltiplos subconjuntos, assegurando que as chaves em um subconjunto são sempre menores que as chaves no subconjunto. Este método visa um balanceamento de carga ideal, onde cada processador recebe uma carga de dados aproximadamente igual para processar e ordenar localmente.

Técnica de Paralelismo

- O paralelismo é implementado através do modelo **MapReduce** no ambiente **Hadoop**. Ambos os algoritmos, Quicksort e Ordenação por Amostragem, utilizam as funções Map para processar os dados de entrada e produzir pares chave-valor, e Reduce para combinar esses valores e produzir um conjunto de dados ordenado.

Linguagem de Programação

- Os algoritmos foram implementados em **Java**, aproveitando as capacidades de gerenciamento de falhas e balanceamento de carga do Hadoop, o que é essencial para processar grandes volumes de dados (Big Data) e executar operações complexas de ordenação em um ambiente distribuído e paralelo.

Avaliação Experimental

- **Ambiente de Teste**: Usou-se um cluster de cinco máquinas com processadores Intel Core 2 Duo, demonstrando a aplicabilidade em sistemas não especializados.
- **Métricas de Avaliação**: Incluem o tempo de execução, speedup (comparativo de tempo de execução entre execuções paralelas e sequenciais idealizadas), e eficiência (uma relação entre o speedup obtido e o número de processadores).
- **Resultados**: O algoritmo de Ordenação por Amostragem mostrou melhor desempenho em comparação ao Quicksort Paralelo, especialmente com o aumento no volume de dados, devido à sua habilidade em manter um balanceamento de carga mais uniforme entre os processadores.