Project Proposal: Distributed Decision Tree

Xialin Liu
Carnegie Mellon University
Pittsburgh, PA 15213
eshang@andrew.cmu.edu

Kedi Xu
Carnegie Mellon University
Pittsburgh, PA 15213
kedix@andrew.cmu.edu

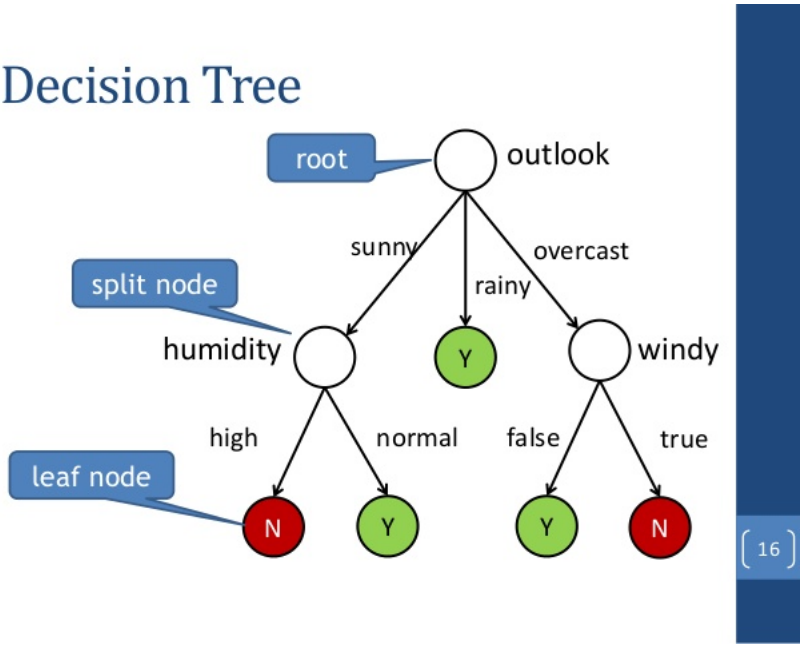
1 SUMMARY

We are going to implement a distributed decision tree on the GHC machine using a hybrid of CPU and GPU, then compare the performance between the sequential version and the parallel version of the algorithm.

link to our webpage: https://xialin0505.github.io/15618Webpage/

2 BACKGROUND

The decision tree is an algorithm for classification, with a tree-like structure in which each branch is the outcome of a test, and a leaf node is the class label. The path from the root to each leaf node is the classification rule of the class.



Below graph 2 is a simplified pseudo-code of decision tree algorithm. Here are several aspects of the algorithm that can benefit greatly from parallelism:

- Splitting on Attributes: The decision tree algorithm evaluates each attribute to determine the best split. This process involves calculating a metric (such as information gain, Gini impurity, or entropy reduction) for each attribute across all examples. Since the calculation

for each attribute is independent of the others, these calculations can be performed in parallel, potentially reducing the overall time to identify the best splitting attribute.

- Handling Subtrees: After selecting an attribute to split on, the algorithm recursively creates new subtrees for each value of the attribute. These subtrees are independent of each other since they operate on disjoint subsets of the dataset. The construction of these subtrees can be parallelized, allowing multiple branches of the tree to be constructed simultaneously. This is particularly beneficial for deep trees or datasets with a large number of categorical values, where the algorithm can branch out significantly.
- Data Partitioning: When splitting the dataset based on the best attribute, the partitioning step can also be parallelized. Each record in the dataset can be independently evaluated to determine which subset (or branch) it belongs to based on the splitting criterion. This process can be efficiently parallelized by distributing the data across multiple processors and performing the partitioning in parallel.

Algorithm 1 Decision Tree Algorithm

procedure DECISIONTREE($Examples, TargetAttribute, Attributes$)
  if $Examples$ are all positive **then return** $positive$
  else if $Examples$ are all negative **then return** $negative$
  else if $Attributes$ is empty **then return** $MajorityValue(Examples)$
  else
    $A \leftarrow$ The Best Decision Attribute from $Attributes$
    $Tree \leftarrow$ A new decision tree with root test A
    for each possible value v_i of A **do**
      $Examples_i \leftarrow$ elements of $Examples$ with $A = v_i$
      if $Examples_i$ is empty **then**
        Add a leaf node with $MajorityValue(Examples)$
      else
        Add the subtree $DecisionTree(Examples_i, TargetAttribute, Attributes - \{A\})$
      end if
    end forreturn $Tree$
  end if
end procedure

3 THE CHALLENGE

Though decision tree is well-studied as a machine learning algorithm, it contains many typical challenges when it comes to parallel implementation. There are many nuances in implementing large-scale parallel training of decision tree. At its core, the decision tree algorithm is sequential because each node split depends on the outcome of the previous one. This sequential dependency can make it difficult to fully parallelize the algorithm, especially in the early stages of tree construction where the choice of the root node and its immediate splits significantly influence the structure of the entire tree.

However, we can still parallelize some intermediate process

- Data Dependencies: When constructing a decision tree, splits at higher levels of the tree create data subsets that are processed independently at lower levels. While processing these subsets can be parallelized, the initial splits have a cascading effect, creating dependencies that dictate the flow of computation
- Memory Access Characteristics: Decision trees often exhibit irregular memory access patterns, especially in datasets with categorical variables leading to variable-sized splits. This irregularity can reduce memory locality and efficiency. Additionally, the need to repeatedly scan the entire dataset for evaluating splits can lead to a high memory access to computation ratio.

- Divergent Execution: The nature of decision tree learning can lead to divergent execution paths, especially in trees with varying depths across branches. This divergence can make workload distribution among parallel processors challenging, as some processors may idle while waiting for others to complete their tasks.
- Workload imbalance: The decision tree is inherently workload imbalance, since the resultant tree might be imbalanced and the prediction process for each input will also vary depending on the depth of the corresponding node of that input. The imbalance work makes it difficult to obtain an ideal speedup when implemented on the GPU.

The constraints of the system are the following:

- Communication Overhead: there are constraints on the communication bandwidth in GPU and DPU if the data to transfer is too large.
- Load Balancing: Ensuring that all processors or nodes in a distributed system have an equal amount of work is challenging, given the variable-sized data subsets created during tree construction. Imbalanced workloads can lead to underutilization of resources and increased computation time.
- Memory Constraints: Decision tree algorithms can be memory-intensive, especially when dealing with large datasets and a high number of attributes. Memory constraints in a distributed system can limit the size of data processed in parallel, requiring careful management of memory resources.

4 RESOURCES

The resource we need for this project is the GHC/PSC machine cluster, with openMP and MPI installed, and GPU core enabled.

We also found some papers discussing the distributed decision tree algorithm, which can be used as a reference.

https://dl.acm.org/doi/pdf/10.1145/2998476.2998478

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8258011

5 GOALS AND DELIVERABLES

The goals are

- **Plan to Achieve**: We are expecting to see a reasonable good speedup between the sequential version and the parallel version of the decision tree.
- **Hope to Achieve**: If the project goes very well, we hope to see a huge improvement (new linear speedup) in our distributed version of the decision tree training algorithm.

The deliverables are some analysis of our parallel implementation of the decision tree algorithm against the baseline. Some profiling results from the machine and the discussion on our results.

6 PLATFORM CHOICE

Our project will use a machine cluster that have GPU enabled, which support inter-core communication frameworks such as openMP and MPI for quickly parallelize a loop when suitable.

7 SCHEDULE

This is the proposed schedule

Week	Plan
3.24	Setup the sequential version of the decision tree and choose the suitable dataset to run the program.
3.31	Finish and Optimize the sequential version of the decision tree, verify the result and use it as the baseline.
4.7	Develop the parallel version of the decision tree algorithm.
4.14	Continue develop the parallel version of the decision tree algorithm
4.21	Collect experimental data and compare the performance. Finish writing the final report.
5.5	Final Project Report Due