

# Making Woody Parallel

## Applying Parallel Computing to decision trees

Hugh McGrade `wbr412@alumni.ku.dk`

Mads Obitsø `scr411@alumni.ku.dk`

Titus Robroek `robroek@di.ku.dk`

-

Department of Computer Science DIKU

November 8, 2018



# Introduction

## Making Woody Parallel

### Introduction

### Working with Woody

Decision tree evaluation  
Extracting from Woody  
Interoperation of Woody and Futhark  
Tree and test data encoding for Futhark

### Writing Futhark

Basic Futhark implementation  
Layered Futhark implementation  
Filtering Futhark implementation  
Treesolver Precompute  
Treesolve Matrix

### Experiment

Experimental Setup  
Results  
Evaluation  
Future Research

### Conclusion

- What is Woody?
- Decision trees and Machine Learning



# Working with Woody

## Decision tree evaluation

### Making Woody Parallel

#### Introduction

#### Working with Woody

##### Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

#### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

Filtering Futhark implementation

Treesolver Precompute

Treesolve Matrix

#### Experiment

Experimental Setup

Results

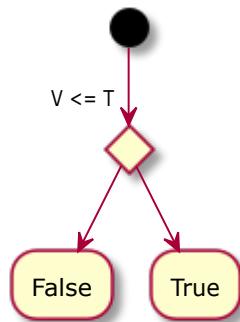
Evaluation

Future Research

#### Conclusion

A decision tree consists of nodes. Non-leaf nodes consist of a conditional check, where:

- **V**: Feature value checked for this node
- **T**: Threshold value of the node that is checked against





# Working with Woody

## Extracting from Woody

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

**Extracting from Woody**

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

#### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

Filtering Futhark implementation

Treesolver Precompute

Treesolve Matrix

#### Experiment

Experimental Setup

Results

Evaluation

Future Research

#### Conclusion

- 1 Python library run against a dataset
- 2 Split data into training and test set
- 3 Fit on training set to create a forest of trees
- 4 Run predictions with test set on this forest



# Working with Woody

## Interoperation of Woody and Futhark

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

Extracting from Woody

**Interoperation of Woody and Futhark**

Tree and test data encoding for Futhark

#### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

Filtering Futhark implementation

Treesolver Precompute

Treesolve Matrix

#### Experiment

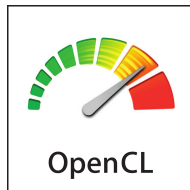
Experimental Setup

Results

Evaluation

Future Research

#### Conclusion



Run as a library or run stand-alone?



# Working with Woody

## Tree and test data encoding for Futhark

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

**Tree and test data encoding for Futhark**

#### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

Filtering Futhark implementation

Treesolver Precompute

Treesolve Matrix

#### Experiment

Experimental Setup

Results

Evaluation

Future Research

#### Conclusion

In order to pass the tree and test data from Woody to Futhark, we encoded each as a series of flat arrays.

- `treeLeftid`
- `treeRightid`
- `treeFeature`
- `treeThres_or_leaf`
  
- `Xtest`
- `nXtest`
- `dXtest`



# Writing Futhark

## Basic Futhark implementation

### Making Woody Parallel

#### Introduction

#### Working with Woody

- Decision tree evaluation
- Extracting from Woody
- Interoperation of Woody and Futhark
- Tree and test data encoding for Futhark

#### Writing Futhark

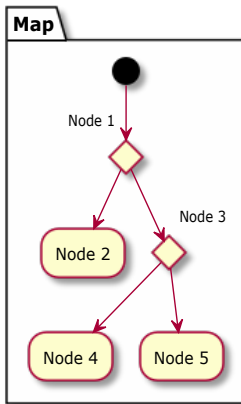
##### Basic Futhark implementation

- Layered Futhark implementation
- Filtering Futhark implementation
- Treesolver Precompute
- Treesolve Matrix

#### Experiment

- Experimental Setup
- Results
- Evaluation
- Future Research

#### Conclusion





# Writing Futhark

## Basic Futhark implementation

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

#### Writing Futhark

**Basic Futhark implementation**

Layered Futhark implementation

Filtering Futhark implementation

Treesolver Precompute

Treesolve Matrix

#### Experiment

Experimental Setup

Results

Evaluation

Future Research

#### Conclusion

```
unsafe map (\ i ->
  let idx = if dindices > 0 then indices[i] else i
  let row_start = idx * dXtest
  in loop node_id = TREE.ROOT.ID
    while treeLeftid[node_id] != TREE.CHILD.ID.NOT.SET do
      if Xtest[row_start + treeFeature[node_id]] <= treeThres_or_leaf[
        node_id]
      then treeLeftid[node_id]
      else treeRightid[node_id]
    ) (iota n_preds)
```





# Writing Futhark

## Layered Futhark implementation

### Making Woody Parallel

#### Introduction

#### Working with Woody

- Decision tree evaluation
- Extracting from Woody
- Interoperation of Woody and Futhark
- Tree and test data encoding for Futhark

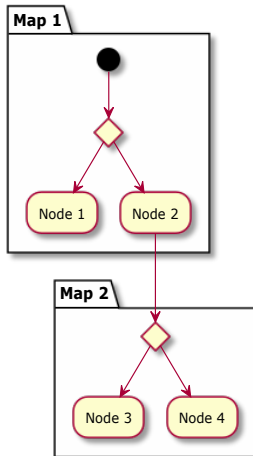
#### Writing Futhark

- Basic Futhark implementation
- Layered Futhark implementation**
- Filtering Futhark implementation
- Treesolver Precompute
- Treesolve Matrix

#### Experiment

- Experimental Setup
- Results
- Evaluation
- Future Research

#### Conclusion





# Writing Futhark

## Layered Futhark implementation

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

#### Writing Futhark

Basic Futhark implementation

**Layered Futhark implementation**

Filtering Futhark implementation

Treesolver Precompute

Treesolve Matrix

#### Experiment

Experimental Setup

Results

Evaluation

Future Research

#### Conclusion

```
loop node_array for row in iota(depth) do
  unsafe map (\ (node_id, data_row_start) ->
    if (treeLeftid[node_id] != 0)
      then (if Xtest[data_row_start + treeFeature[node_id]] <=
        treeThres_or_leaf[node_id]
        then treeLeftid[node_id]
        else treeRightid[node_id])
      else node_id)
    (zip node_array data_row_starts)
```



# Writing Futhark

## Filtering Futhark implementation

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

#### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

**Filtering Futhark implementation**

Treesolver Precompute

Treesolve Matrix

#### Experiment

Experimental Setup

Results

Evaluation

Future Research

#### Conclusion

- Layered provides a structured way of iterating over layers
- Many passes end early
- These are still considered by the maps



# Writing Futhark

## Filtering Futhark implementation

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

#### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

**Filtering Futhark implementation**

Treesolver Precompute

Treesolve Matrix

#### Experiment

Experimental Setup

Results

Evaluation

Future Research

#### Conclusion

```
let node_array = replicate n_preds 0
let is_not_leaf = (\ (node_id, _) -> treeLeftid[node_id] != 0)
let is_leaf = (\ (node_id, _) -> treeLeftid[node_id] == 0)
let next_node = (\ (node_id, data_row_start) ->
  ((if (is_not_leaf (node_id, data_row_start)) then (if Xtest[
    data_row_start + treeFeature[node_id]] <= treeThres_or_leaf[
    node_id] then treeLeftid[node_id] else treeRightid[node_id])
  else node_id), data_row_start))

let nodes = zip node_array data_row_starts
let leaves = []
let (_, leaves) = loop (nodes, leaves) for row in iota(depth) do
  let new_nodes = (unsafe map next_node nodes)
  in (unsafe filter is_not_leaf new_nodes, leaves ++ (unsafe filter
    is_leaf new_nodes))

let result = map (\ (a, _) -> a) (radix_sort_by_key (\ (_, a) -> a) i32.num_bits i32.
  get_bit leaves)
in result
```



# Writing Futhark

## Treesolver Precompute

### Making Woody Parallel

#### Introduction

#### Working with Woody

- Decision tree evaluation
- Extracting from Woody
- Interoperation of Woody and Futhark
- Tree and test data encoding for Futhark

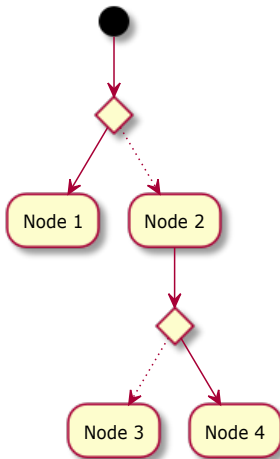
#### Writing Futhark

- Basic Futhark implementation
- Layered Futhark implementation
- Filtering Futhark implementation
- Treesolver Precompute**
- Treesolve Matrix

#### Experiment

- Experimental Setup
- Results
- Evaluation
- Future Research

#### Conclusion





# Writing Futhark

## Treesolver Precompute

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

#### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

Filtering Futhark implementation

**Treesolver Precompute**

Treesolve Matrix

#### Experiment

Experimental Setup

Results

Evaluation

Future Research

#### Conclusion

```
let next_node
  (row: [] f64)
  ((left, right, feature, thres) : (i32, i32, i32, f64)) : i32 =
  if row[feature] <= thres then left else right
```

```
let make_next_tree
  (tree: [] (i32, i32, i32, f64))
  (row : [] f64) : [] i32 =
  map (next_node row) tree
```

```
let traverse
  (next_nodes: [] i32) : i32 =
  let (last, current) = (0, next_nodes[0])
  let (result, _) = loop (last, current) while current != 0 do (current, next_nodes
    [current])
  in result
```

```
let nodes = zip4 treeLeftid treeRightid treeFeature treeThres_or_leaf
let rows = unflatten nXtest dXtest Xtest
```

```
let next_nodes = unsafe map (make_next_tree nodes) rows
```

```
in unsafe map traverse next_nodes
```



# Writing Futhark

## Treesolve Matrix

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

#### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

Filtering Futhark implementation

Treesolver Precompute

**Treesolve Matrix**

#### Experiment

Experimental Setup

Results

Evaluation

Future Research

#### Conclusion

```
let repeated_criteria = flatten (replicate nXtest treeFeature)
let repeated_offsets = flatten (map (\ i -> replicate treelength i) (steps 0 nXtest
    dXtest))
let flcr = map2 (+) repeated_offsets repeated_criteria
let scattered_features = unsafe map (\ i -> Xtest[i]) flcr
let threshold_result = map2 (<=) scattered_features (flatten (replicate nXtest
    treeThres_or_leaf))
let left_or_right = (\ b l r -> if b then l else r)
let repeatedLeft = flatten (replicate nXtest treeLeftid)
let repeatedRight = flatten (replicate nXtest treeRightid)
let directions = map3 left_or_right threshold_result repeatedLeft repeatedRight

in unsafe map traverse (unflatten nXtest treelength directions)
```





# Experiment

## Experimental Setup

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

#### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

Filtering Futhark implementation

Treesolver Precompute

Treesolve Matrix

#### Experiment

Experimental Setup

Results

Evaluation

Future Research

#### Conclusion

- Comparing with Woody
- Only the covtype dataset.
- Troubles with swig: Measurements may be impacted

## Command to measure futhark code:

```
cat tmp_tree | ./futharkProgram -t measurements/futharkProgramTimes -r 10 >
measurements/futharkProgram.txt
```

## Measuring Woody:

```
times = []
for i in range(n):
    cpu_pred_start_time = time.time()
    cpu_pred = super(WoodClassifier, model).predict_single_tree(Xtest)
    cpu_pred_stop_time = time.time()
    micro = int((cpu_pred_stop_time - cpu_pred_start_time) * 1000000)
    times.append(micro)
return times
```





# Experiment

## Results: Varying train data size

### Making Woody Parallel

#### Introduction

#### Working with Woody

- Decision tree evaluation
- Extracting from Woody
- Interoperation of Woody and Futhark
- Tree and test data encoding for Futhark

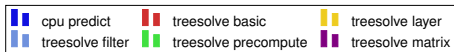
#### Writing Futhark

- Basic Futhark implementation
- Layered Futhark implementation
- Filtering Futhark implementation
- Treesolver Precompute
- Treesolve Matrix

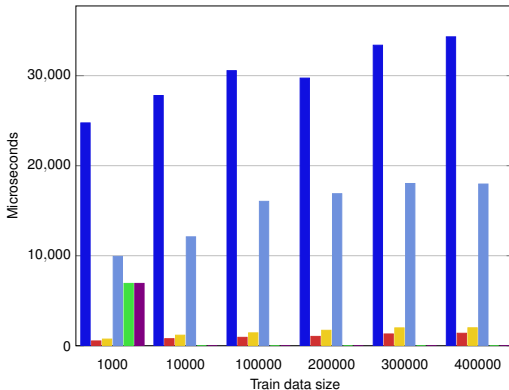
#### Experiment

- Experimental Setup
- Results**
- Evaluation
- Future Research

#### Conclusion



Microseconds used predicting a single tree as average of 10 runs





# Experiment

## Results: Varying number of predictions

### Making Woody Parallel

#### Introduction

#### Working with Woody

- Decision tree evaluation
- Extracting from Woody
- Interoperation of Woody and Futhark
- Tree and test data encoding for Futhark

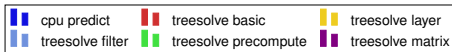
#### Writing Futhark

- Basic Futhark implementation
- Layered Futhark implementation
- Filtering Futhark implementation
- Treesolver Precompute
- Treesolve Matrix

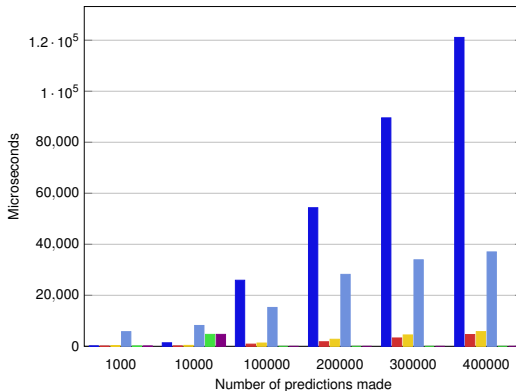
#### Experiment

- Experimental Setup
- Results**
- Evaluation
- Future Research

#### Conclusion



Microseconds used predicting a single tree as average of 10 runs





# Experiment Evaluation

## Making Woody Parallel

- Basic and layer are fast.
- GPU code scales better with bigger tests

### Introduction

### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and  
Futhark

Tree and test data encoding  
for Futhark

### Writing Futhark

Basic Futhark implementation

Layered Futhark  
implementation

Filtering Futhark  
implementation

Treesolver Precompute

Treesolve Matrix

### Experiment

Experimental Setup

Results

**Evaluation**

Future Research

### Conclusion



# Experiment

## Future Research

### Making Woody Parallel

#### Introduction

#### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

#### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

Filtering Futhark implementation

Treesolver Precompute

Treesolve Matrix

#### Experiment

Experimental Setup

Results

Evaluation

**Future Research**

#### Conclusion

- Matrix and Precompute might be promising
- GPU code scales better with bigger tests



# Conclusion

## Making Woody Parallel

### Introduction

### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

Filtering Futhark implementation

Treesolver Precompute

Treesolve Matrix

### Experiment

Experimental Setup

Results

Evaluation

Future Research

### Conclusion

We have proposed a number of approaches to parallelising the evaluation of decision trees using Futhark. Our findings show that as a whole this parallelisation is promising for the performance of decision tree evaluation on large datasets.



## Making Woody Parallel

### Introduction

### Working with Woody

Decision tree evaluation

Extracting from Woody

Interoperation of Woody and Futhark

Tree and test data encoding for Futhark

### Writing Futhark

Basic Futhark implementation

Layered Futhark implementation

Filtering Futhark implementation

Treesolver Precompute

Treesolve Matrix

### Experiment

Experimental Setup

Results

Evaluation

Future Research

### Conclusion



Belgiu, Mariana, and Lucian Drăguț. "Random forest in remote sensing: A review of applications and future directions." ISPRS Journal of Photogrammetry and Remote Sensing 114 (2016): 24-31.



Gieseke, Fabian, and Christian Igel. "Training Big Random Forests with Little Resources." arXiv preprint arXiv:1802.06394 (2018).