### 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 ar Futhark

Tree and test data encodir for Futhark

### Writing Futhark

Basic Futhark implementation

implementation

Eiltoring Eutha

Transalus Deservation

Treesolve Matri:

### Experimen

Evanrimontal Catu

Results

-----

0---!--

■ What is Woody?

■ Decision trees and Machine Learning



### Working with Woody Decision tree evaluation

Making Woody Parallel

Introductio

Working with Woody

Decision tree evaluation

Extracting from Woody

Futhark

Tree and test data encoding for Futhark

### Writing Futhar

Basic Futhark implementation Layered Futhark

Filtering Fuths

implementation

Treesolve Matri:

### Experimen

Experimental Se

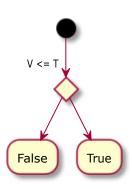
Evaluation

Future Resea

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 Wo

Tree and test data encodi

for Futhark

### Writing Futhar

Basic Futhark implementation

implementatio

Filtering Futha

Treesolver Precomp

Treesolve Matri

### Experimen

Experimental Set

Results

\_ - - -

- Python library run against a dataset
- Split data into training and test set
- 3 Fit on training set to create a forest of trees
- 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

for Futhark

### Writing Futhark

Basic Futhark implementation Lavered Futhark

implementation

Filtering Futhar

Treesolver Precompt

Treesolve Matri

#### Experimer

Evanrimontal Cat

Results

-----

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 evaluati Extracting from Wood

Interoperation

Tree and test data encoding

### Mission Futbook

Basic Futhark implementation

Layered Futhark

implementation

implementation
Transplant Processory

Treesolve Matri:

### Experimer

Result

Evaluation

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
- dXt.est.



## Writing Futhark Basic Futhark implementation

### Making Woody Parallel

### Introduction

### Working with Woody

Decision tree evaluation

Extracting from Woody

Futhark

Tree and test data encoding for Futbark

### Writing Futhark

#### Basic Futhark implementation

Layered Futha

implementation

Filtering Fuths

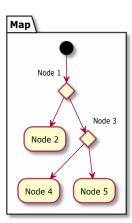
Transactions December

Treesolve Matri

### Experimen

Experimental Setur

Results





# Writing Futhark Basic Futhark implementation

### Making Woody Parallel

### Introduction

### Working with Woody

Decision tree evaluation

Extracting from Wool

Tree and test data enco

### Writing Futhark

#### Basic Futhark implementation

Layered Futhark

implementation

Implementation

Transaction Matei

### Experimen

Evanrimontal Catu

Results

....



## Writing Futhark Layered Futhark implementation

### Making Woody Parallel

### Introduction

### Working with Woody

Decision tree evaluation

Extracting from Woody

Futhark

Tree and test data encoding

### Writing Futhark

Basic Futhark implementation

#### Layered Futhark implementation

implementation

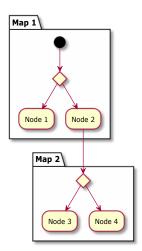
Filtering Fuths

Treesolver Precomp

### Treesolve Matri

Results

Evaluation





### Writing Futhark Layered Futhark implementation

### Making Woody Parallel

### Introduction

### Working with Woody

Decision tree evaluation

Interoperation of Woody a

Futhark

Tree and test data encoding

### Writing Futhark

Basic Futhark implementation

### Layered Futhark

implementation

Treesolver Precomo

\_ .

#### Experimen

Experimental Seti

Results

\_ . . . .



## Writing Futhark Filtering Futhark implementation

### Making Woody Parallel

### Introduction

### Working with Woody

Decision tree evaluation

Interoperation of Woody a

Tree and test data enci

### Writing Futhark

Basic Futhark implementation Layered Futhark

#### Filtering Futhark implementation

Treesolver Precompu

### Evporimor

Results

⊏valuation

0---!--

 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

### ....

### Working with Woody

Extracting from Woody

Futhark

for Futhark

### Writing Futhark

Basic Futhark implementation Layered Futhark

### implementation Filtering Futhark

implementation
Treesolver Precomp

### Evnerimen

Experimental Setu

Results

Future Researc

```
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, ) \rightarrow a) (radix_sort_by_key (\ (, a) \rightarrow 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

Futhark

Tree and test data encoding

### Writing Futhark

Basic Futhark implementation

implementation

.....

#### Treesolver Precompute

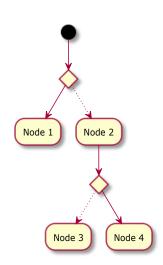
Treesolve Matrix

### Experiment

Evnorimental Satur

Results

Litaliani





## Writing Futhark Treesolver Precompute

### Making Woody Parallel

Introduction

Working with Woody

Decision tree evaluation

Interoperation of Woody a

Tree and test data end for Futback

Writing Futhark

Basic Futhark implementation Layered Futhark

Filtering Futhar

Treesolver Precompute

\_\_\_\_

Experiment

Results

Evaluation

Canalusian

```
let next node
  (row: [1f64)
  ((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

### Making Woody Parallel

### Introduction

### Working with Woody

Decision tree evaluation

Extracting from Woo

Futhark

for Futhark

### Writing Futhark

Basic Futhark implementation Layered Futhark

Filtering Futhark

Treesolver Precomput

### Treesolve Matrix

### Evnerimen

Experimental Setu

Results

E 4 ... D ....

Conclusion

```
let repeated_criteria = flatten (replicate nXtest treeFeature)
```

- let flcr = map2 (+) repeated\_offsets repeated\_criteria
- let scattered\_features = unsafe map ( $\setminus$  i  $\rightarrow$  Xtest[i]) flcr
- let left\_or\_right = (\ b | r -> if b then | 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

Interoperation of Wo

Tree and test data encodi

### Writing Futhar

Basic Futhark implementation Layered Futhark

Filteriae Futhe

Implementation
Treesolver Precompu

IT COSOTVO IVIALITA

### Experimen

Experimental Setup

Evaluation

\_ . . .

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

### Introductio

### Working with Woody

Decision tree evaluatio

Interoperation of Woody Futhark

Tree and test data encoding

### Writing Futhark

Basic Futhark implementation

implementatio

Filtering Fuths

Treesolver Precomp

Treesolve Matri

#### Experimen

Experimental Setu

### Results

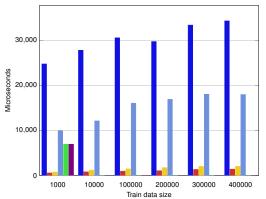
\_ . . . .

\_ . . .

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

Tree and test data enci

### Writing Futhark

Basic Futhark implementation Lavered Futhark

implementation

Filtering Futhark implementation

Trooppolyo Matri

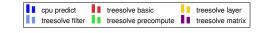
#### Experimen

Experimental Setu

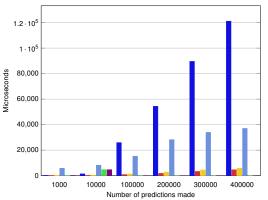
### Results

Eutura Passar

Conclusion



### Microseconds used predicting a single tree as average of 10 runs







### Experiment Evaluation

### Making Woody Parallel

### Working with Woody

### Writing Futhark

Basic Futhark implementation

Results

Evaluation

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



## Experiment Future Research

### Making Woody Parallel

### Introduction

### Working with Woody

- ...

Extracting from Woody

Futhark
Tree and test data encoding

### Writing Futhark

Basic Futhark implementation

Layered Futha

implementation

Implementation

Transalus Matrix

### Experimen

Experimental Setu

Results

Future Research

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



### Conclusion

### Making Woody Parallel

Introduction

Working with Woody

Decision tree evaluation

Interoperation of Woody

Tree and test data enco for Futhark

Writing Futhar

Basic Futhark implementation

implementation

Filtering Futhark implementation

Treesolve Matri:

Experimen

Evnerimental Setu

Results

E to Book

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

Extracting from Woody
Interoperation of Woody and

Tree and test data encodi for Futhark

### Writing Futhark

Basic Futhark implementation Layered Futhark

implementation

implementation

Transalus Matri

### Experimen

Experimental Setu

Results

Evaluation

Conclusion

## Belgiu, Mariana, and Lucian Drăguţ. "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).