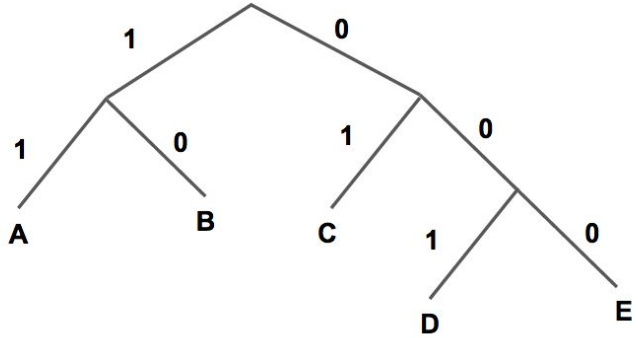
Parallelizing Huffman Encoding / Decoding



Malcolm Mashig (mjm2396)

COMS 4995: Parallel Functional Programming
Columbia University, Fall 2021

Huffman Coding Overview

1) Creation of a code tree based on character likelihood/frequency
2) Encode each character in the text via traversal of the tree
3) Decode bit sequences via traversal of the tree

Purpose: optimize compression ratio by giving small codes to frequent characters

Compression Ratio: (# of bits in compressed text) / (# of bits in original text)

…as I define it

Parallelization

1. Split input text into batches, then encode in parallel
2. Split encoded bit sequence into batches, then decode in parallel

Caveats:

- What about creating the code tree?
  - Should we create a separate code tree per batch?
  - How do we store the code tree so that it can be independently known during decoding
- How do we split encoded bit sequence into batches?
  - **Character codes vary in length, and so our batches may split up individual characters**

Approach

Example Parallel Huffman Encoding File:

Number of Batches:	4
First Index:	828
Second Index:	1657
Third Index:	2469
Encoding:	00000000110001111011111011100011101101010001111000011001101000101100110111101110101011000010110011001111111100001001100010010010100010001000110101010100010000111011010100001111111110…

Indices are determined by generating a cumulative sum of the bit sequence lengths during encoding

Implementation

```
let decInds = getInds dec_instr
let decBits = getBits dec_instr
let decBatches = batchBits decBits decInds
let decodings = Prelude.map (decode tree) decBatches `using` parList rdeepseq
let decoded = concat decodings
```

Small-Scale Test

ORIGINAL TEXT:

But there is a problem. To decode the message, you need the coding tree for that message. I'm sure there are a variety of techniques to transmit the tree as part of the message. One of the things Apfelmus did that intrigued me was develop a stack based language and interpreter to represent the morse code tree as a string of characters. I think this would make a fine way to encode the tree into the front of the message. We should be able to extract a string of characters from the Huffman tree for a particular message and use that string of characters to reconstruct the tree on the other side. Now this will add some significant overhead to the output for a given message, but I'm not necessarily concerned with space efficiency here as much as learning about these kinds of transformations.

ENCODED:

00000000110001111011111011100011101101010001111000011001101000101100110111101110101011000101100110011111111000010011000100100101000100010001101010101000100001110110101000011111110010101010101011110110
10011011101000011000001011111100011110100100010000011101101010000011010101010001100111110110110001110011111001000001001010101110011101101010101111000111111100101010101011110110100110000100110111011110
00110111111100101011111001111000011101101010001111000010110111100001011001101110010110111110011001110011000000010101001001110100011011101011110110011100010000111110100101000111001010011100111101111 1
10101011111111001111000111011010100001110011110010000101110100011011111011011111100001010100100111011010100001111111001010101010111101101001100001001100011101111101000001010100100111011010100001110 1
10101100111110110110101000110001100111011110100110001100111111111111010100001000110010100000111011010101111000110011110111001111100111011011110100010000011111110000110000010111010000100010011011 1
00100011001010111011110010110010101110101101101110001010001100011011101010001000000110011011111101101101111101011110110100001011111100100000110011111011101000111101111011110011101000111001110010100
01111001101111011110010101001111011100011101101010000111111010101111010100000110101010100010000111001111001000010111010001011001010111001111100111101101100001010100100011011101010110111101101101111
01000111101011000010011011101100111011010110011111011000101001110110101100110100011000000101111110011001010000011111110111100010110000101100010011100111110100001100000101101100000011100101001001111 0
01101010101000100001110110101000011100111100100001100111110111001010011101101010000010010111010111110111000010101001001110110101000011111110010101010101111011010011000010011000100111000010101101001 0
11111100110010100000001100011000010110110001011001100001110010100100110001000111100111101101101111000101100101011100111110011111011011000010101001000110111010101101111011011011110100011110100001001 01
11010111111100111011010100001100011111111110010010100111111110111111000111001111001000001001010101110010110011011110110111110110010110111111001100110110111001111111001010101010111101101000010111 11
10010000011111010101010000111011010101011111000101011100111110011111011011000010101001000110111010101101111011011011110100011110100011100101000111100011010101111010101011100111111110011011110001110110101
00001110011111001000001011111000111011010100000101111011010100011100101011001010001001100001001100011100010111000000001110110101100110100011000001100101100101100100101101000010000010100101111111100001
01011001110110111101100101001110010110110111111011110000101110111001000111110101001011010000011100101001110110101000000101111110111011011111111110111000010010101011100101100110110110011101110010011110 0
01111111001010101010111101101001101110100001100011111101110001101110111100011011111110011110010111100011110100011011001010101010110111110010110010110000000110101011110011011000111111010001000001 10
00001100111101101000101011011111011011011000010001001010011100101101110011001111001101011000000011010100011110000101110100011111111110011011101000101110100001100110010110111111011001111101101100010
1101100010101111110111000111011010100101010000110001011100111111001000101000010101001001110011110111111010100100101010111111111101111101100101011111010101100001110001100 0

DECODED:

But there is a problem. To decode the message, you need the coding tree for that message. I'm sure there are a variety of techniques to transmit the tree as part of the message. One of the things Apfelmus did that intrigued me was develop a stack based language and interpreter to represent the morse code tree as a string of characters. I think this would make a fine way to encode the tree into the front of the message. We should be able to extract a string of characters from the Huffman tree for a particular message and use that string of characters to reconstruct the tree on the other side. Now this will add some significant overhead to the output for a given message, but I'm not necessarily concerned with space efficiency here as much as learning about these kinds of transformations.

LOSSLESS?
True

Large-Scale Test

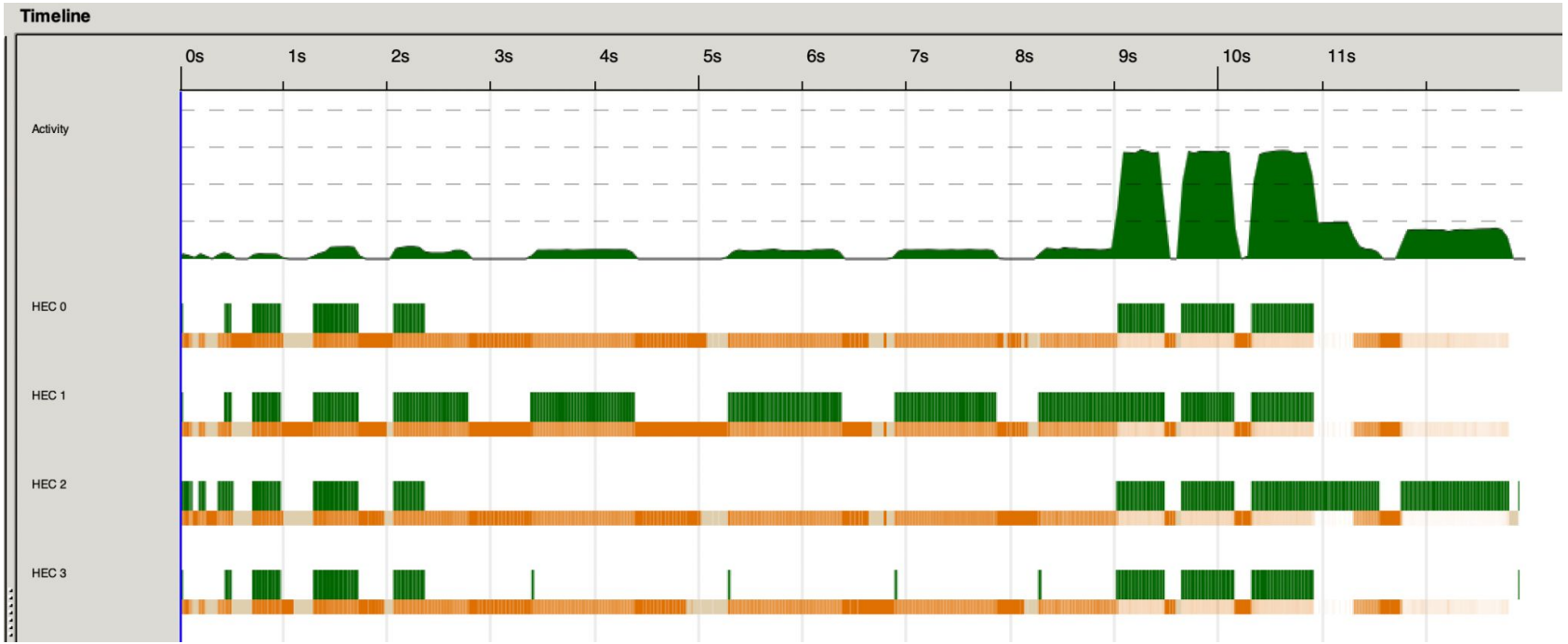
Shakespeare Works: http://www.gutenberg.org/files/100/100-0.txt

Length: 5,716,512 characters

Compression Ratio: ~0.60

real 18.236s (100 batches)
real 16.432s (500 batches)
real 16.064s (2000 batches)
real 13.146s (10000 batches)

Threadscope



What I’ve Learned + Next Steps

- Still investigating improvements to achieve speedup
- Compression is sequential by nature, and parallelization is difficult

Next steps:

- Attempt other parallelization strategies and compare