Aabash 2018B4A70887P

Design:

a) Huffman algo is not parallelized because

→ Based on literatures available online, the overhead of parallelizing is high.

→ main reason is that our input space is from sparse space – printable ascii characters. Also, ~~since~~ if this assumption is not true, even then the ~~need~~ symbols are bounded by 256 because splitting is done based on the bytes.

For files ≥ 10 mb, the time for running huffman, given the frequency of each character, is very low.

b) For encoding,

i) File is opened using MPI-open and each process gets its own copy of data, which is distinct and complete data is separated b/w processes. Data is divided on byte level and ~~for start index~~ every process gets no. of bytes $\in \left\{\left\lfloor \frac{size}{P} \right\rfloor, \left\lceil \frac{size}{P} \right\rceil\right\}$

→ Almost uniformly distributed for large files.

Time complexity $= O\left(\frac{n}{P}\right)$,

ii) Each process calculates frequency of each of byte sequences. → $O\left(\frac{n}{P}\right)$ time

iii) We need cumulative frequency, so we do All reduce on vectors of size 256 across all processes. Now, size of this vector ≤ 256

⇒ overhead is $O(\log P)$

iv) All processes will get total frequency and use them to construct Huffman tree → ≤ 256 characters

→ $O(1)$ time

(This is repeated calculation → serial part this is not parallelized)

(v) Encoding is done by simple linear traversal
→ $O(\frac{n}{p})$ time

(vi) File is stored → $O(\frac{n}{p})$ time

vii) File offsets → MPIScan → $O(\log p)$

⇒ Overall time complexity = $O(\frac{n}{p})$

↳ Serial time is only for construction of Huffman tree (because calculation is repeated).
while rest part is ~~for~~ parallelized completely.

Since overhead of transmission is ignored in calculations. ($n >> p$) and msg size is fixed

(c) Similar analysis can be done for decoding to get $O(\frac{n}{p} + 2\log p)$ as total time.

Part b: To calculate the value of f, following method is used:

① T_1' & T_p' are defined.
→ Since only serial part, effectively, is construction of Huffman tree, T_1' is User + SYS time for construction of tree and rest is in T_p'
→ Calculated in code

② Let $\sigma(n)$ be time for sequential part of code
$\phi(n)$ — parallel — when executed sequentially

⇒ $T_1' = \sigma(n)$
$T_p' = \frac{\phi(n)}{p}$

③ By definition, T_1 = time to run code sequentially

$$= \sigma(n) + \phi(n)$$
$$= T_1' + p\,T_p'$$

T_p = time to run code parallely

$$= \sigma(n) + \frac{\phi(n)}{n}$$
$$= T_1' + T_p'$$

④ By definition,

$$\text{efficiency} = \frac{T_1}{T_p} \leftarrow \text{calculate in code.}$$

⑤ for fraction of code running sequentially,

$$\text{efficiency} = \frac{T_1}{T_p} = \frac{1}{f + \frac{(1-f)}{p}} = \frac{T_1' + p\,T_p'}{T_1' + T_p'}$$

$$\Rightarrow \frac{T_1' + p\,T_p'}{T_1' + T_p'} = \frac{1}{f + \frac{(1-f)}{p}}$$

$$\Rightarrow \frac{\sigma(n) + \phi(n)}{\sigma(n) + \frac{\phi(n)}{p}} = \frac{1}{f + \frac{(1-f)}{p}}$$

Comparing, $f = \frac{\sigma(n)}{\sigma(n) + \phi(n)} = \frac{T_1'}{T_1' + p\,T_p'} = \frac{T_1'}{T_1}$

for encoding a file of 1 mB,

efficiency (avg) = 0.003

for decoding, avg efficiency = 0.004

Part c

Time complexity → derived already as $O(\frac{n}{p})$ for both.

Speed up = $\frac{T_1}{T_p} = \frac{O(n)}{O(\frac{n}{p})} \sim O(p)$

Efficiency = $\frac{\text{Speedup}}{p} = \frac{O(p)}{p} \sim 1$.

→ almost no overhead for large input

Cost = $pT_p = p \cdot O(\frac{n}{p}) = O(n)$

→ This is optimal

Ans (c)

Time complexity → derived already $= O(\frac{n}{p} + 2\log p)$

Speed up = $\frac{T_1}{T_p} = \frac{n}{\frac{n}{p} + 2\log p}$

Efficiency = $\frac{\text{Speedup}}{p} = \frac{n}{n + 2p\log p}$

Cost = $pT_p = p(\frac{n}{p} + 2\log p) = n + 2p\log p$.

for $n >> p$, $\sim O(n)$

For isoefficiency,

$E_1 = \frac{n_1}{n_1 + 2p_1 \log p_1}$ & $E_2 = \frac{n_2}{n_2 + 2p_2 \log p_2}$

For isoefficiency,

$$E = \frac{1}{1 + \frac{2p \log p}{n}}$$

$$\Rightarrow 1 + \frac{2p \log p}{n} = \frac{1}{E}$$

$$\Rightarrow \frac{2p \log p}{n} = \frac{1}{E} - 1$$

$$\Rightarrow n = \frac{2p \log p}{\frac{1}{E} - 1} \Rightarrow n \propto 2p \log p$$

$\Rightarrow$ Iso Efficiency function : $w = 2Kp \log p$

(d) optimal no. of processors:

$$T_p = \Theta\left(\frac{w}{p}\right) = \Omega\left(\frac{w}{f^{-1}(w)}\right)$$

where $w = \Theta(f(p))$

Here, $w = f(p) = 2Kp \log p$

$$\Rightarrow p \approx \frac{w}{\log w} \Rightarrow T_p = \Omega\left(\frac{w \log w}{w}\right)$$

$$= \Omega(\log w)$$