PUNE INSTITUTE OF COMPUTER TECHNOLOGY
DHANKAWADI, PUNE

HIGH PERFORMANCE COMPUTING MINI-PROJECT REPORT
ON

"GENERIC COMPRESSION - RUN LENGTH ENCODING CONCURRENTLY ON MULTI−CORE. "

SUBMITTED BY

Pritesh Nikale 41453
Sunveg Nalwar 41466

Under the guidance of
Prof. Amruta Chandorkar

DEPARTMENT OF COMPUTER ENGINEERING
Academic Year 2021-22

Contents

1 Problem Statement

Perform Run length encoding concurrently on many core GPU. Run-length encoding is a form of lossless data compression in which runs of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run.

2 Abstract

Multi-core programming is increasingly used for acceleration of data-parallel functions, such as image transforms and motion estimation. However, the entropy coding stage is typically executed on the CPU due to inherent data dependencies in lossless compression algorithms. We propose a simple way of efficiently dealing with these dependencies, and present a parallel compression algorithm specifically designed for efficient execution on many-core architectures. By parallelization of lossless compression algorithms not only do we significantly speed-up the entropy coding, but we also enable completion of all encoding phases using OpenMPI.

3 INTRODUCTION

In the Run Length Encoding, we need to compute the indexes of the elements to be stored and their codes i.e. the length of the symbol run. The first approach for computing the codes is based on the use of another parallel primitive, the reduction, for counting the symbol or summing up the number of times a symbol appeared in its run. Further analysis of dependencies in the original algorithm resulted in a method which does not depend on the architectural support for the atomic operations. Instead of summing the number of occurrences for each symbol in parallel, the indexes of last elements in each of the run are determined on the basis of the flags. These index values then can be used to compute the total number of elements that appear in between these locations without actually counting the occurrences. The resulting count corresponds to the number of times an element appeared in its run.

4 OBJECTIVE

- To implement a generic compression algorithm using parallelism.
- To compare the speed-up and efficiency of serial and parallel implementation of run-length encoding compression algorithm.

5 Scope

This parallel compression algorithm can be easily combined with the already available GPU-accelerated image transforms and motion estimation algorithms, to enable execution of all codec components directly on GPUs. With this approach we speed-up the entropy coding by 5-20x, and reduce the data transfer time from the GPU to system memory.

6 Test Cases

```
Files :
input1 => average case
input2 => worst case
input3 => best case
```

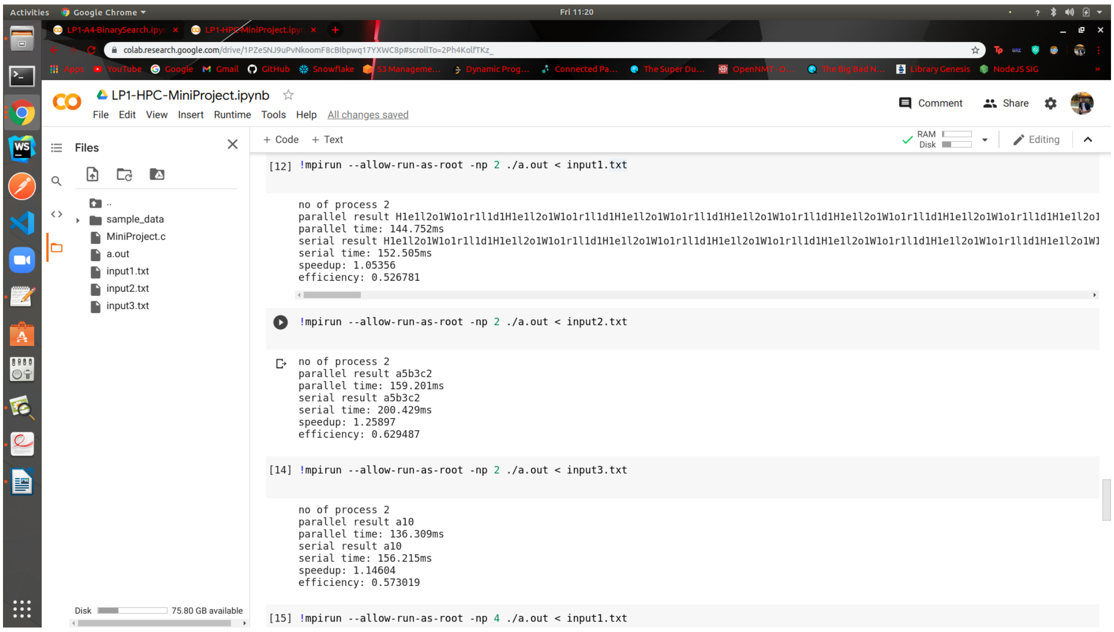

Figure 1: Output with no. of processes 2

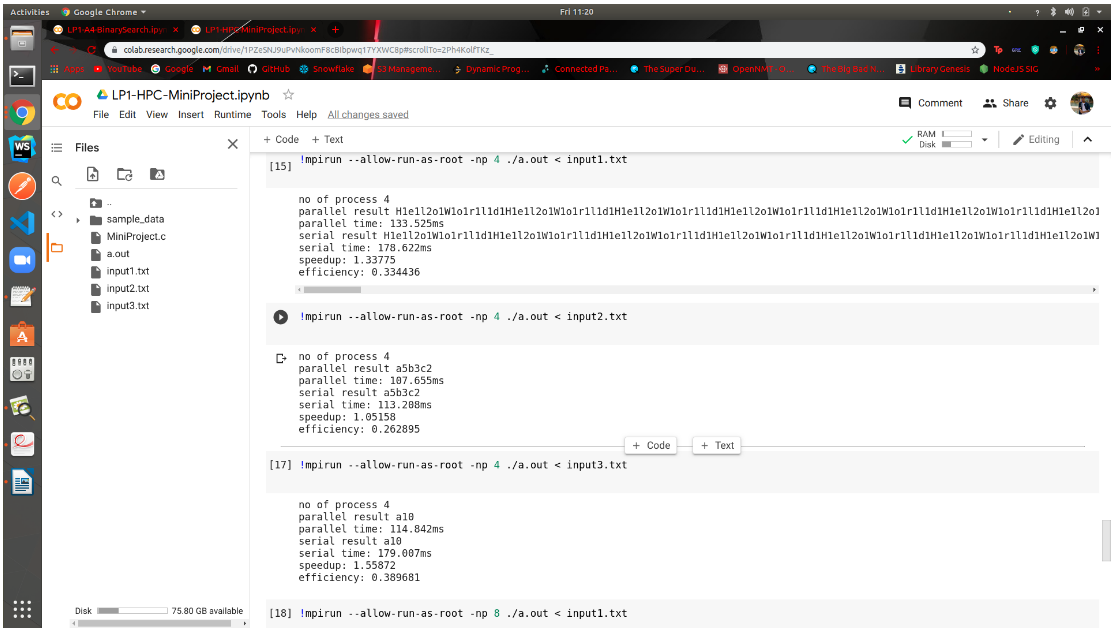

Figure 2: Output with no. of processes 4

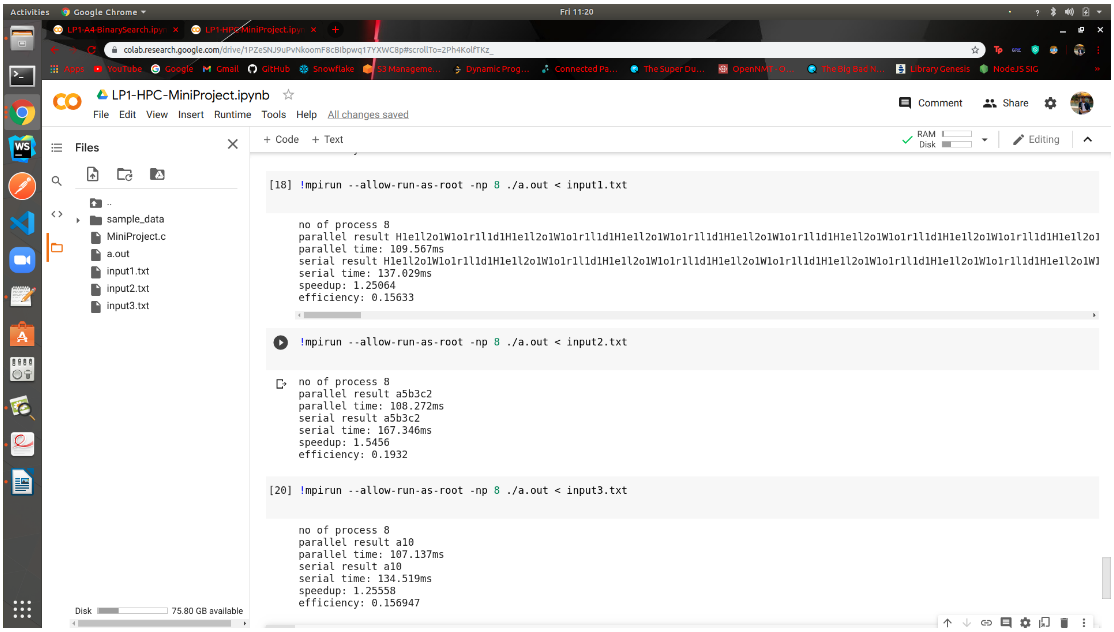

Figure 3: Output with no. of processes 8

7 Result

Highest speedup and efficiency is given when 4 process are used for all i.e best case, worst case and average case.

8 Conclusion

We presented parallel lossless compression designed for efficient execution on many core architectures supporting parallelism. The parallelization of the Run-Length encoding resulted in processing rates up to 2x speedup.

References

[1] https://www.geeksforgeeks.org/run-length-encoding/

[2] https://stackoverflow.com/questions/31890523/how-to-use-mpi-gathe

[3] https://www.open-mpi.org/doc/v4.0/