PCP Course Project

by khalid shareef

Submission date: 23-Dec-2020 10:47PM (UTC+0530)
Submission ID: 1480914045
File name: Report_1.pdf (69.15K)
Word count: 836
Character count: 4271

CS5300 - Parallel and Concurrent Programming

Wait Free sFCM

Sowrya Gali - CS18BTECH11012
Khalid Shareef - CS18BTECH11029

Introduction

In this project, we attempt to provide a wait-free implementation of the sFCM algorithm. The sFCM algorithm is widely used in MRI image segmentation and other medical applications. This project is done as part of the Parallel and Concurrent Programming Course under Dr. Satya Peri's guidance.

Prior Knowledge

The membership matrix calculation for each data point (pixel). To understand our implementation, first, we need to know how a general sFCM algorithm works. Here is a brief explanation of the same :

In the sFCM algorithm, we select the number of clusters into which the image is to be classified. We randomly choose c data points from the input matrix and assign them as initial means of c clusters. Then, based on these means, calculate membership (without considering the spatial information) of each data point with each of the clusters using the formula :

$$\mu_{ij}\,(\texttt{membership of ith point to the jth cluster}) = 1/\left(\sum_{k=1}^{c}(d_{ij}/d_{ik})^{2/m-1}\right)$$

where,

d_{ij} represents the euclidean distance of ith data point from the jth cluster

c represents the number of clusters

m is fuzziness parameter(index) $\in [1, \infty)$

Then, recalculate the new means using the calculated membership values using the formula :

$$v_i \text{ (mean of the ith cluster)} = (\sum_{j=1}^{N} \mu_{ij}{}^{m} x_j)/(\sum_{j=1}^{N} \mu_{ij}{}^{m})$$

We started off our implementation by declaring some essential matrices like “updating” and “updated”(of type atomic bool), which represent the status of a data point whether its membership values are calculated or not. The importance of these is shown later. Then we start defining the sFCM function which, for a given submatrix (region) of the input image(matrix), calculates the membership of all the data points in that submatrix to each of the clusters. This sFCM function is run by multiple threads on different disjoint submatrices(regions) of the input image(matrix). The spatial information is incorporated using a spatial function is given by :

$$h_{ij} = \sum_{k \in NB(x_j)} \mu_{ik}$$

where,

NB is a square window(5*5) centered at point x_j

Our Approach

We developed two approaches for this task both focus on workload distribution among the threads. In the first approach, we partition the image equally among the threads and each thread calculates the membership values first for their respective regions. Then after completion of all threads, new threads respawned to calculate spatial values using these membership values and updates the membership matrix according to it.

In the second approach, we developed a novel way of using faster threads to help slower threads for the calculation of initial membership values. Then the spatial values are calculated from these values and at the end, the clusters are updated.

We developed this helping strategy using atomic variables. The threads after calculating the membership values for its region start to calculate the spatial part. The pixels at the border of regions have a window extending into the other regions, for which threads may not have yet calculated the values. So this thread helps in calculating the values without waiting for them to complete.

Before calculating some data point's membership values, the thread first checks if the updating and updated values of that datapoint are false(i.e indicating that no other thread is calculating the membership of this datapoint and no thread has already calculated its membership). Each thread computes the membership of all the data points in its region. When calculating the spatial matrix(value of the spatial function at each data point) at the boundary points (for which the NB crosses its boundary) of a submatrix (for a thread), if the membership of the points on the other regions(of other thread) is not calculated yet, then this thread should wait for the other thread to calculate these membership values. To avoid this and to make this thread calculate the value of membership of the data point of another thread ('helping' other thread), we use the updated and updating matrices and calculate that value.

Implementation Details

We implemented these algorithms in C++. The images are fed in the form of a matrix in a text file, which can be retrieved easily with a simple python program. The clustering happens on this matrix which is taken as input.

For checking the updates we used "updated" and "updating" atomic variables and using we implemented the sFCM algorithm with the proposed strategy 2.

Experiments

We have done some preliminary experiments with 2 to 4 threads, results show that initially, simple load distribution performs better than the helping strategy. But we could not extend

it to more threads due to some bugs in our memory management code. We will extend it soon to show promising results.

Conclusion

The strategy we proposed has its root in many other “helping” strategies developed in the class like snapshots or non-blocking linked list implementations. We will refine the algorithm properly so that it works for many threads and with proper memory management.

PCP Course Project

ORIGINALITY REPORT

2%	0%	2%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Sudha Tiwari, S. M. Ghosh. "Chapter 141 Segmentation Techniques Using Soft Computing Approach", Springer Science and Business Media LLC, 2020 Publication	1%
2	Submitted to SASTRA University Student Paper	1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		