Lab Course: Distributed Data Analytics
Exercise Sheet 2

Prof. Dr. Dr. Schmidt-Thieme, Daniela Thyssens
Information Systems and Machine Learning Lab
University of Hildesheim

Submission deadline: Friday May 13, 23:59PM (on LearnWeb, course code: 3116)

Instructions

Please following these instructions for solving and submitting the exercise sheet.

1. You should submit two items a) a zipped file containing python scripts and b) a pdf document.
2. In the pdf document you will explain your approach (i.e. how you solved a given problem), and present your results in form of graphs and tables.
3. The submission needs to be made before the deadline, only through learnweb.
4. Unless explicitly stated, you are not allowed to use scikit, sklearn or any other library for solving any part. All implementations must be done by yourself.
5. In each lab you have to show your solution works for $P = \{2, 4, 6, 8 \ldots\}$ and provide the timing results (whether it is stated in the question or not)

Exercise 1: Point to Point Communication (10 Points)

In this exercise you will write a parallel program using point-to-point communication routines. Suppose a worker with rank 0 has an integer array and it wants to send to all $P-1$ workers in COMM_WORLD, lets call this routine **sendAll**.
Short description of a naive way *NsendAll*:
A naive way to send this array is using a *for* loop at worker 0 and sequentially send it to all other processes i.e. it will take $P-1$ steps. **Hint:** Make sure all the workers exit the sendAll routine at the same time i.e its useful to use
MPI_Barrier at the end of this function.

Short description of an efficient way: *EsendAll*
Another possible ways is to use a *recursive doubling* algorithm, which will require $log(P)$ steps.
Suppose you have P workers, where $P \geq 2^d$ i.e. if $P = 33$ than $d = 5$ and rank is the current worker ID. Lets say the root worker has rank 0. The root worker sends to worker with Rank 1 and worker with Rank 2 only. All other workers first receive a message from recvProc, i.e.
$recvProc = int((rank-1)/2)$
and sends to two more processes
$destA = 2 \times rank + 1$ and $destB = 2 \times rank + 2$.
But before sending, make sure $destA$ and $destB$ exist. To make sure every process has finished put a MPI.Barrier() just before returning from *sendAll* routine.

Your tasks are to write:

1. Implement the *NsendAll* routine using the naive way i.e using a single for loop.
2. Implement the *EsendAll* routine using the efficient way as explained above.
3. Compare performance of the two implementations by recording the time to finish each task. You can fix the number of process $P = 2^d$ i.e. 16 and 32, and send an array of size 10^3 or 10^5 or 10^7. [Hint: see which of the array sizes fit in your memory and even if you have less cores you should still use 16 and 32].

Exercise 2: Collective Communication (10 Points)

In this exercise you have to find an Image histogram. Images generally have RGB or gray scale values. Finding histogram is just calculating the frequency of occurrence of each gray scale or RGD value. You have to provide parallel implementation using collective routines only.
Note: You have to implement frequency calculation and you should not use opencv or other build-in methods for frequency calculation.
Along with the code, please explain the following:

1. Pick an image with a high resolution i.e. resolution $\geq$ 2048x2048.
2. Data division strategy i.e. how you divide your data among processes.
3. How you assign tasks to different processes?
4. How you combine results from all the processes?
5. Did you implement for RGB or gray scale histogram? (2 points if it works with both)
6. Provide runtime analysis on varying number of processes i.e. $[1, 2, 3, 4]$. You have to show that your solution actually is reducing time to calculate histogram, if you add more processes. (3 points)

[Note:] If you dont know how to read an image in python, please see Annex below. (Install, use opencv and histogram tutorial)

Annex

1. Palach, Jan. Parallel Programming with Python. Packt Publishing Ltd, 2014.
2. For reading an image as gray scale or RGB you can use OpenCV.
   (a) Install opencv: conda install -c menpo opencv
   (b) A simple tutorial `http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_image_display/py_image_display.html`
   (c) Histogram of an image in opencv `http://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_image_histogram_calcHist.php`
3. To time your program you have to use MPI.Wtime(): `http://nullege.com/codes/search/mpi4py.MPI.Wtime`
4. MPI tutorial (C/C++): `https://computing.llnl.gov/tutorials/mpi/`