Class Activity

You are given a large grayscale image of size 10000 × 10000 pixels and a 3×3 convolution filter kernel (e.g., edge detection or blur). You must apply this filter to every pixel in the image (excluding the border pixels). You are asked to design a parallel program that performs this convolution operation efficiently using multiple threads. As a parallel programming designer, to make better design decisions, answer the following questions, clearly stating all your assumptions.

(a) What type of partitioning technique would you use?
(b) How will you assign work to processors?
(c) What data must be communicated between processors and when and how often communication will happen?
(d) Estimate the amount of communication per processor.
(e) How will you ensure load balancing? Image can have non-uniform complexity (e.g., parts are more textured than others).

Solution:

(A) Row-wise data partitioning. Each thread gets contiguous block of rows. Less communication overhead then 2D partitioning.
(B) Rows will be equally divided between processors. (10000-2)/P, excluding the first and last rows.

(C) Each thread needs one extra row above and one below its assigned rows to compute valid 3×3 convolution at its edges, so each thread must read one row from the previous and one from the next processor's partition (except for the first and last thread). This must happen once before computation begins if the image is read-only and not modified during convolution.
(D) Assuming one byte per grayscale pixel, and a total of 10000 pixels (10000 bytes), each processor sends 1 row above and 1 row below. So total communication is approximately 20,000 bytes per processor.
(E) Static scheduling with row partitioning, or dynamic scheduling using work-stealing or hybrid approach where image is divided into smaller tiles (e.g., 100x100) and allow threads to dynamically pick tile to process.