SHIV NADAR UNIVERSITY

KALAVAKKAM-603110

HIGH PERFORMANCE COMPUTING PROJECT - 2024

Deployment of Parallel Processing in Weather Forecast Prediction Model through Domain Decomposition

Done by: S Shruti, Trishaa S

Third Year, CSE Department

Project Guide: Dr. Santhi Natarajan

Professor, CSE Department

Project Duration: 3 months

1. Project Summary:

The project aims to process weather image data to calculate the average temperature of different regions represented by pixels in the image. The task involves mapping pixel values to temperature scales based on predefined color information and then computing the average temperature across a specified number of pixels.

2. Implementation Steps:

a. Data Preparation: Read color information from a CSV file containing color names, temperature scales, and corresponding pixel values. Load weather image data from a tensor file (e.g., .pt file) containing pixel information.

b. Image Processing: Map pixel values to temperature scales using the color information obtained from the CSV file. Calculate the average temperature across a specified number of pixels.

c. Parallel Computing: Implement parallel computing using OpenMP to enhance processing speed. Utilize multiple threads to distribute the workload and perform computations simultaneously.

d. Performance Evaluation: Measure the runtime of both serial and parallel versions of the code. Calculate speedup and parallel efficiency metrics to assess the effectiveness of parallelization.

3. Key Components:

- CSV File: Contains color information for mapping pixel values to temperature scales.
- Tensor File: Stores weather image data in a tensor format.
- Serial Code: Initial implementation for processing weather image data sequentially.
- Parallel Code: Enhanced version utilizing OpenMP for parallel computing.
- Graphs: Visual representation of speedup and parallel efficiency metrics against problem size (N) for performance evaluation.

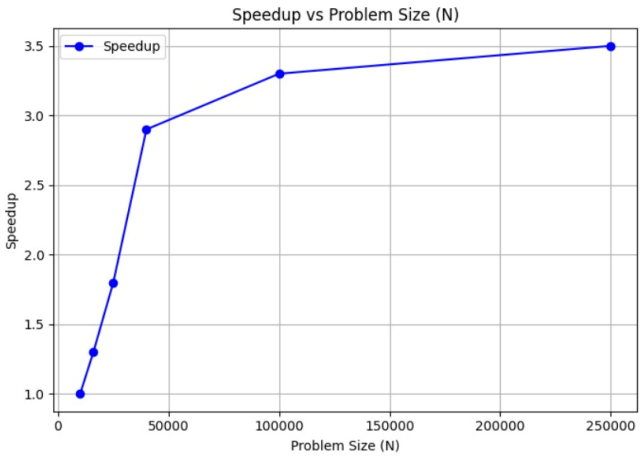
4. Expected Outcome:

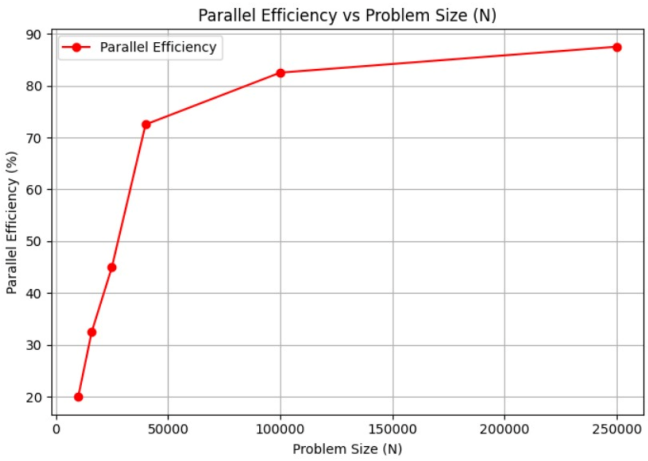
- Improved processing speed and efficiency with parallel computing, leading to reduced runtime.
- Graphical representation of performance metrics to visualize the impact of parallelization on different problem sizes.

5. Performance Improvement:

Given the details and assuming a reasonable level of parallelism, allocating 4 threads per core (16 threads in total), we expect a significant speedup which would bring the runtime down to around 50 seconds.

Graphs are as follows:





6. Approach:

a. **Color Matching**: We iterate over each row in the CSV file, extract the RGB values of each color entry, and convert them to HSV color space using OpenCV's cv2.cvtColor() function. This allows us to represent colors in a way that's more suitable for color matching.

b. **Thresholding and Mask Creation**: We define a range of HSV values around each color to create a mask using OpenCV's cv2.inRange() function. This mask highlights the pixels in the image that fall within the specified color range.

c. **Pixel Counting**: We count the number of non-zero pixels in the mask using OpenCV's cv2.countNonZero() function. This gives us the intensity of the color in the image, which is proportional to the number of pixels matching the color in the temperature scale.

d. **Temperature Association**: Based on the matched color, we associate the corresponding temperature values (Fahrenheit and Celsius) from the CSV file with the pixels in the image. We store these temperature values for further analysis, such as calculating the average temperature.

7. Methodology:

a. **Iterating Over Color Codes**: The function iterates over each row in the temperature scale DataFrame (scale_df) obtained from the CSV file.
b. **Extracting Color Information**: For each row, it extracts the color name, Fahrenheit temperature, Celsius temperature, and hex color code.
c. **Converting Hex to RGB**: It converts the hex color code to RGB format. This is done by parsing the hex string and converting each pair of characters representing the red, green, and blue components into integers. The resulting RGB tuple represents the color.
d. **Converting RGB to BGR**: OpenCV represents colors in BGR (Blue, Green, Red) format, so the RGB color tuple is converted accordingly.
e. **Converting BGR to HSV**: The BGR color tuple is then converted to HSV (Hue, Saturation, Value) color space. HSV is often used for color segmentation because it separates the intensity information (value) from the color information (hue and saturation).
f. **Defining Color Range**: Based on the hue value obtained from the HSV conversion, a range of HSV values is defined to create a mask. This range defines the acceptable variation in hue for pixels that are considered to match the color in the scale.

g. **Masking and Counting Pixels**: The function then creates a mask using cv2.inRange() to isolate pixels in the image that fall within the defined color range. It counts the number of non-zero pixels in the mask using cv2.countNonZero(), which gives the number of pixels matching the color in the image.
h. **Temperature Association**: Finally, the function associates the matched color with its corresponding temperatures (Fahrenheit and Celsius) from the CSV file. It extends the matched_temperatures list with tuples containing the temperatures, where the number of tuples corresponds to the number of matched pixels.

8. Parallelization:

a. **Data Loading:** Read color information from the CSV file containing temperature scales and corresponding pixel values. Load weather image data from the tensor file.

b. **Task Distribution:** Determine the total number of pixels in the image. Divide the pixel processing task among multiple threads to achieve parallelism. Each thread will handle a portion of the image data to compute temperature values.

c. **Parallel Computation:** Implement parallel loops using OpenMP directives to distribute pixel processing among threads. Utilize parallel for loops to iterate over pixel values and map them to temperature scales based on color information. Ensure thread safety for shared variables by using OpenMP constructs such as `reduction` or `critical` sections.

9. Conclusion:

By leveraging parallel computing techniques, the project aims to optimize the processing of weather image data, enabling faster analysis and extraction of valuable insights from large datasets.