Distributed Photo Collage

Distributed Computing Project 2021/2022

Francisco Teixeira 84843

Broker.py

- This component acts on the folder containing the photos to resize and merge; it distributes tasks and communicates with the workers to which it is connected, via UDP socket:

  - Its initial arguments are the folder containing the pictures, and the desired height of the final collage;
  - The UDP socket is initialized at the internally defined address;
  - The broker waits for messages; when a worker connects, tasks are given to it;
  - The folder with the images is sorted by the modification date of each image;
  - The image folder is analyzed to find out whether there are any images with the desired height at the start - depending on the result, tasks are distributed to two different queues - one for resizing and the other for pasting;
  - The resizing tasks are given to the workers (request message), to guarantee that the images will have the correct height; when they all meet this requirement, they can be merged;
  - When there is only one collage of all the initial images, the process is finished - the broker communicates this to all the workers so that they can shut down. Some statistical information about the process is then printed out. The broker also shuts down.

Worker.py

- This component is responsible for receiving and executing the tasks given by the broker, communicated via UDP socket:
    - Its initial argument is the broker's address;
    - The worker connects to the broker by UDP socket;
    - The worker waits for messages; when a broker sends a task (i.e. a request message), it is processed and sent (reply message).
    - If the collage is completed, the worker will receive a message indicating this, and can then disconnect.

Message Protocol

- The broker and the various workers exchange JSON messages with each other in order to coordinate which operations have to be performed, and creating the possibility to exchange information with each other:

Message Structure	Goal
{ "***type***": "*request*", "***request***": "*resize*", "***height***": *height* }	Sent from a broker to a worker, when the broker wants a worker to resize an image. The worker has access to the height set in the broker at the start of the program;
{ "***type***": "*request*", "***request***": "*merge*" }	Sent from a broker to a worker, when the broker wants a worker to merge two images;
{ "***type***": "*reply*", "***task***": "*resize*" }	Sent from a broker to a worker, when the broker wants a worker to merge two images;
{ "***type***": "*reply*", "***task***": "*merge*" }	Sent from a worker to a broker, it serves as a response to the request made for the merge task.

Message Structure	Goal
{ "***type***": "*status*", "***status***": "*ready*", "**id**": address }	Sent from a worker to a broker, it indicates that a connection has been made and that the worker is ready to receive tasks. The broker has access to the worker's address;
{ "***type***": "*status*", "***status***": "*finish*"	Sent from a broker to a worker, it indicates that the broker's program has finished;
{ "***type***": "*info*", "***info***": "*title*", "**title**": title }	Sent whenever an image is exchanged between broker and worker, and vice versa.

Utilities.py

- This component has several auxiliary functions that are used both in the broker and in any worker, as well as internally:

  - ***send_img*** *to send images: returns its name information through a JSON message;*
  - ***recv_img*** *to receive images: receives the image in a folder created in the worker's directory as the port number of its address, or in the home folder in the case of the broker; the function returns the path to the image;*
  - ***resize_img****: resizes the image;*
  - ***merge_img****: merges two images together; the name of the new image is random (combination of characters and numbers), and saved in PNG format; the function returns the path to the image;*
  - ***new_directory****: creates a new directory; is used in the recv_img function;*
  - ***addr_format****: returns a formatted address;*
  - ***get_height****: returns the height of an image;*
  - ***timeout****: creates a wrapper that is used to indicate when a function takes too long to execute; has a time limit until the timeout occurs.*

Observations

- All communication between workers and brokers is done over UDP, whether it is JSON messages or exchanging images;
- Both the broker and the workers print out the task to be performed and a warning when it is complete;
- When a broker receives a resized image, it is named the same as it was before it was sent; in the case of a merge, the name is random - to avoid conflicts - so it will be necessary to delete the two original images;
- In the worker's case, the folder with the received and created images is only deleted when the whole process is finished;
- A broker can distribute tasks to several workers iteratively, thus distributing the load - despite this, the implemented solution does not aim at increasing performance due to the number of workers;
- A worker can timeout and the broker has this information, but cannot resume the process;
- A broker can timeout and the worker has that information; then, it shuts down.