Assignment 1 2INC0 2014

Robin van der Veer 0779303
Robert van Eerdewijk …

1 Overview

There were a 3 main design decisions we had to make:

- What form of interprocess communication to use
- How to structure the Job and Result messages
- How to let the workers know when they can quit

Besides those three aspects, we will also shortly give a short analysis of the results we obtained.

2 What form of interprocess communication

At first we did not realise that we had to use the message queue system, so we started thinking about which form of interprocess communication would be best suited for this application. We though that named pipes would be easiest and most reliable, because they are not restricted in their size, while message queues are. We also shortly considered sockets, but decided it would be a bit overkill.
We did run into some problems while using names pipes that we did not encounter in our later message queue-base version. The main problem with the named pipes was that the farmer would not continue reading from the pipe once there were no more workers on the other side. This ment that after the workers were done with their real work, they had to give the farmer some time to read the last few results from the pipe before terminating. Because we did not like the approach of just letting the workers sleep for a certain amount of time, we had to use a third pipe, which was used to let the workers know when it was safe to close all pipes and terminate.
After we had made this version, and found out that we had to use a message queue, it was not much trouble to update the piped-version to use message queues instead. As said, this version is (at least in out opinion) a bit more elegant, because the farmer can continue to read from the queue even after all workers terminated.

3 How to structure the job and result messages

This one also ties in to the next point: *How to let the workers know when they can quit,* as will become apparent. The job message that the parent places in the queue, should have at least two fields: the x and the y coordinate of the pixel that is to be calculated*. Similarly for the result message, it also contains the x and y pixel coordinates of the pixel for which is message is ment.
The other field of the result message was an easy choice: the color of the pixel. Since there is no more information that has to be communicated from the worker to the farmer, the result message is now complete.

However, the farmer does have some more information to tell the worker: when to quit. The worker is in no way aware of how many pixel are still remaining, and thus it has no way of, by itself, knowing when it can stop pulling messages from the queue.
We decided that the easiest way to let the worker know when it can quit, it by sending one additional field in the message: the command field. In our application it only has two possible values, although extension (up to a total of 256) can be made if one would so desire. It is represented a single char, which in our application takes on either the value ‘j’ or the values ’s’. If the worker sees an ’s’ job message, it does the work described by the job (calculating the pixel values), and than terminates. If it sees a ‘j’, is does the job, and than start another one.
The farmer writes only ‘j’ messages, expect for the last NROF_WORKERS pixel, where it instead markers the messages with an ’s’. This guarantees that all workers will stop when it is time.
We could have also hacked this command into the message by just setting the 31th bit of the x or y coordinate to 1 for a stop job (since such large values are never reached in practice any way). However we felt this was a bit to ‘hacky’ and not very elegant.

4 Results

We will now very shortly note some results that we found while doing tests.
First of all, we did not find any noticeable difference in our pipe based vs our message queue base versions.
Secondly, we found that the version with 1 worker and 10 messages performed a lot better than the version with 32 workers and 1 message. We think this is because in the version with 32 workers, time is shared by a larger number of processes. This means that although every single process might only get to pull 2/3 messages from the queue before a context switch occurs, all 32 of them will drain the queue before the farmer will have had time to fill it back up again. On the other hand, if there is only 1 worker, chances are pretty good that the farmer will have enough cpu times to prevent the queue from being depleted by that single worker, and thus the worker and work more efficiently.

*We decided that the worker should do the conversion from pixel coordinates to mandlebrot coordinates. We made this decision because if we convert the pixel coordinates (int) to the mandlebrot coordinates (double) in the farmer, than the worker or the farmer has to make the conversion back, than we cannot (because of limited double precision) be sure to retrieve the same pixel coordinates again. Which would have ment we had to do all kinds of rounding to retrieve the correct coordinates.