MP6 Report

Xu Zihan 3220110781 zihan1.22@intl.zju.edu.cn

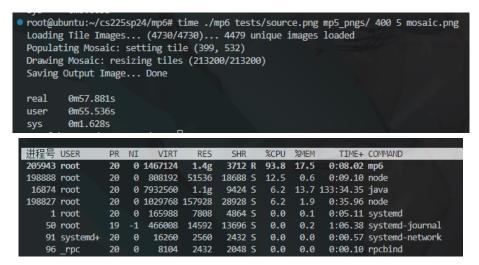


Figure 1 Performance of original MP5

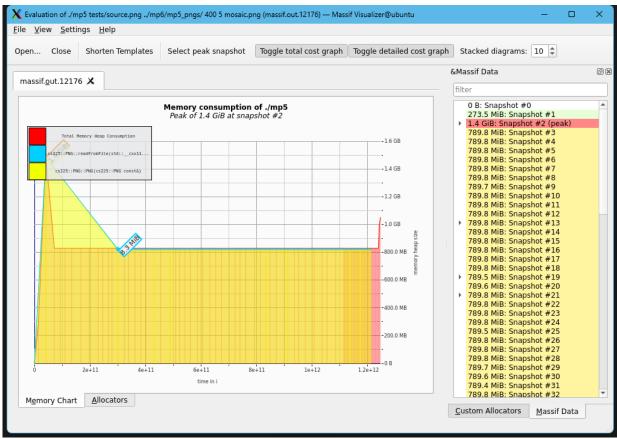


Figure 2 Memory consumption of original MP5

Reduce Space Complexity:

Analyzing memory usage using Valgrind's Massif tool, it can be seen that all png files are loaded into memory.

In the original version, main.cpp::vector<TileImage>getTiles (string tileDir) initialized each image as a TileImage and returned a vector of TileImage, which is the sum of all tiles, greatly wasting memory space.

- 1. My operation changes its return value to the map<Point<3>, string>of the file name and corresponding color value. As is well known, the size of the string is much smaller than that of the tileImage containing PNG.
- 2. Rewrite the TileImage class simultaneously and move the function and value that calculates AverageColor to

main.cpp. This can also reduce the complexity of mapTiles (Source Image const&TheSource, map<Point<3>, string>&TheTiles).

- 3. Once FNN finds nearby color blocks, using the TheTiles index, it can directly return the file name at the corresponding position on the canvas.
- 4. To this end, modify the function maptiles{cpp,h}::string* get_match_at_idx(const KDTree<3>&tree, map<Point<3>, string>&TheTiles, const Source Image&TheSource, int row, int col) and mosaiccanvas {cpp, h}:: const string&MosaicCanvas::getTile (int row, int column).
- 5. Finally, when rendering the canvas, load all tiles one by one and call the modified tiles. Theoretically, tiles will only occupy a maximum of 1 * w * h of memory space.

Time: 57.881s -> 2m16s

Memory: 1.4G -> 233.1M

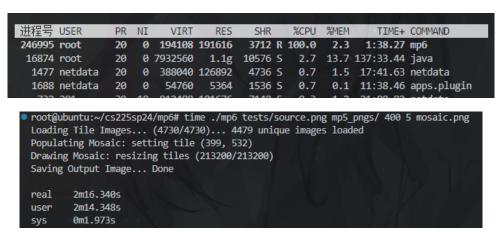


Figure 3 Performance of space-optimized MP5

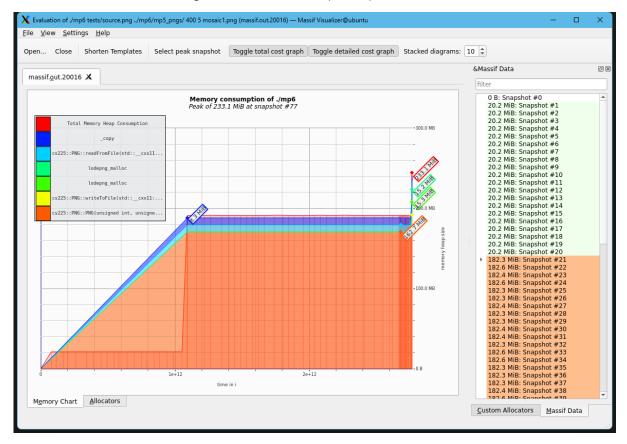


Figure 4 Memory consumption of space-optimized MP5

Reduce Time Complexity:

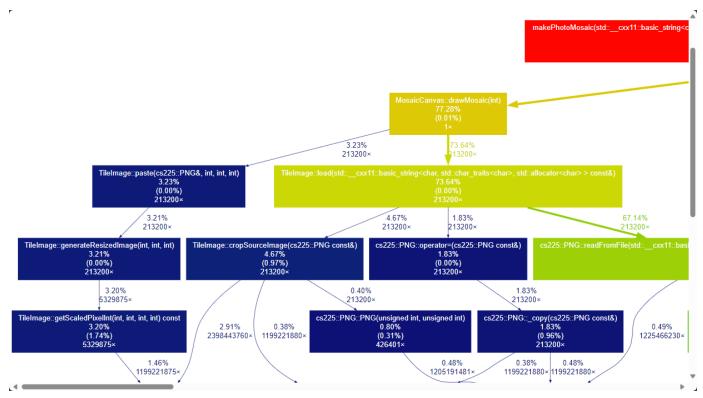


Figure 5 Part of The Call Graph of Functions in Space-optimized MP5 and Its Time Cost

To see which function is taking up a lot of time, using gprof to check program performance, it was found that MosaicCanvas::drawMosaic(int) takes up a lot of time.

The reason is that its way of traversing each coordinate causes the same tile to be loaded repeatedly (possibly thousands of times), resulting in the time complexity of O (w * h * w '* h'), where w 'and h' are the width and height of tiles.

After modification, use map<string *, vector<tuple<int, int>>>, and index the location where this image is used with the image file name. When traversing, each image only needs to be loaded once, and after loading, the positions used are traversed, filling the canvas at each corresponding position. The time complexity is O (w * h+w '* h'), which is much smaller than the previous value.

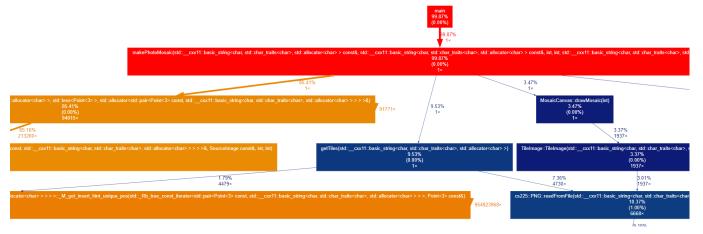


Figure 6 Part of The Call Graph of Functions in MP6 and Its Time Cost

Upon rechecking the performance, it can be seen that MosaicCanvas::drawMosaic(int) only takes up 3.8% of the time, even better than the MP5 before spatial optimization. And the mapping process came back to its original major role in time consumption. The completion time of the entire program also reached 38 seconds, which is

significantly faster than the original MP5, and there is no additional cost in memory.

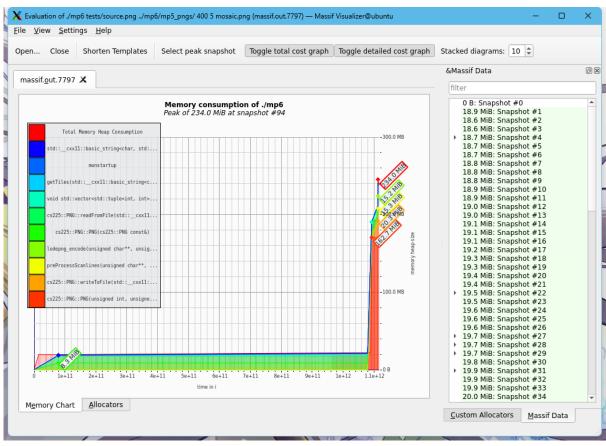


Figure 7 Memory consumption of final MP6

real	0m36.956s	
user	0m34.339s	
sys	0m2.469s	

Final Performance:

Memory: 1.4G -> 233.1M ->234.0M Time: 57.881s -> 2m16s-> 36.956s