CONTACT NAME
Akiyoshi Wakatani: wakatani@konan-u.ac.jp

Improved GPGPU Implementation of Fractal Image Coding with Quadtree Decomposition

Akiyoshi Wakatani, Akio Murakami (Konan University, JAPAN)

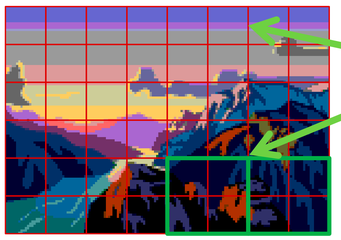
Summary: GPGPU (General Purpose computing on Graphic Processing Unit) attracts a great deal of attention, which is used for general-purpose computations like numerical calculations as well as graphic processing. In this paper, we implement adaptive fractal image coding algorithms on GPUs by using CUDA (Compute Unified Device Architecture) in order to achieve a given quality of a compressed image and evaluate the improved version of the coding algorithm by enhancing the occupancy of GPU cores.

Background and motivation

Fractal image coding

- The most similar domain block should be found for each range block.
- Good compression rate
- But, too much computation cost

- To achieve a high performance computation by using GPU power
- Especially, we try to implement adaptive Fractal image coding algorithm efficiently.



parallel algorithm 1

```
parallel for (i=0;i<N; i+=16)
 parallel for(j=0;j<N; j+=16)
  if(encode(i,j,16)>criteria)
    done(i,j,16);
parallel for (i=0;i<N; i+=8)
 parallel for(j=0;j<N; j+=8)
  if(done[i][j]!=1)
    if(encode(i,j,8)>criteria)
      done(i, j,  8);
parallel for (i=0;i<N; i+=4)
 parallel for(j=0;j<N; j+=4)
  if(done[i][j]!=1)
   encode(i,j,4);
```

By using index vectors

parallel algorithm 2

```
for (R=16;R>=4;R/=2){
 for(i=0,k=0; i<N; i+=R)
  for(j=0; j<N; j+=R)
   if(R==16||is_done(I,j,R*2)!=1)
     vvec[k++]=i;
 nn=k;
 parallel for (i=0;i<nn; i++)
  if(encode(vvec[i], R)>criteria)
   done(vvec[i], R);
}
```

The elements of the index vector vvec contain the location of the range block that must be compressed with the following range block size.

The global memory access of range[vvec[i]] increases the access cost because this memory access is not a coalesced communication. However, since the global memory access to range[vvec[i]] can be done out of the most inner loop and the size of the most inner loop is rather large, the increase ratio of the global memory access to the index vector is relatively small and thus can be ignored.

By enhancing the occupancy of GPU cores

However, since the summation for each range block is carried out by one thread, the occupancy of GPU cores does not seem to be enough. So, the improved version should calculate it by plural threads. Although the number of threads in a thread block can increase to enhance the occupancy, the communication between the threads within in the thread block may increase the overhead ("parallel program 3").

Each CUDA core on the GPGPU environment (SIMD) is in charge of one loop element, so a core that is in charge of the done block is not executing anything. Thus, the efficiency of the parallelism does not seem to be excellent on the GPGPU environment..

Results and discussion

The parallel program 2 is about 2 to 3.5 times faster than the parallel program 1

As the PSNR criterion increases, the speedups of the parallel program 2 over the parallel program 1 decrease.

Why?

The reason is that as the PSNR criterion increases, the computational ratio of 4×4 and 8×8 range blocks increase and thus the difference between the parallel programs 1 and 2 decreases

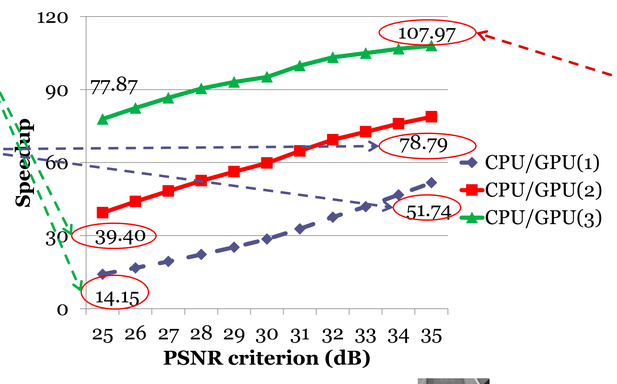


Fig.1 Results 1 (Lena)

GPGPU system : PC with Intel Core i5-3470 CPU and Nvidia GTX Titan under CentOS 6.4.

The performance gain of the parallel program 3 at the small PSNR criterion (e.g. 25 dB) is larger than that at the large PSNR criterion (e.g. 35 dB).

Why?

The reason is that the improvement of the parallel program 3 is done for the 16x16 range block size and the computation ratio of 16x16 range block size is relatively larger when the PSNR criterion is small.

The value of the speedup for the Boat image is larger than that for the Lena. The reason is that the speedup of the GPU over the CPU with the 16x16 range block size is smaller than other range block sizes and the computational ratio of 16x16 range block size for the Boat image is relatively smaller than that for the Lena image, so the Boat image achieves a large speedup.

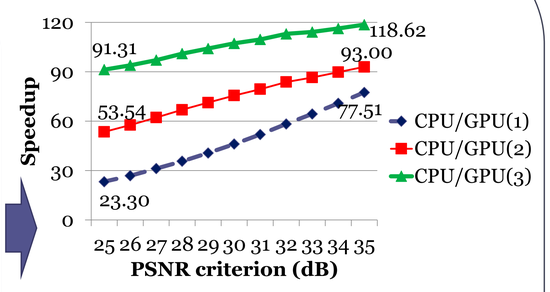


Fig.2 Results 2 (Boat)

Conclusions: We implement adaptive fractal image coding algorithms on GPUs by using CUDA. Our results for the Lena and Boat images show that the memory access overhead of the index vector does not affect the superiority of the parallel I program 2 over the parallel program 1 and the performance gain of the parallel program 3 at the small PSNR criterion is larger than that at the large PSNR criterion. We achieves the speedup of up to 118.62 using GTX Titan.

Contact:
Akiyoshi Wakatani (Konan University, Faculty of Intelligence and Informatics)
Email: wakatani @konan-u.ac.jp FAX +81-78-435-2540