Komprese/dekomprese JPEG obrázků pomocí 3D akcelerační karty

Lucie Matušová xmatus21@stud.fit.vutbr.cz
Jan Wozniak xwozni00@stud.fit.vutbr.cz

1 Úvod

Úkolem projektu bylo vybrat částí algoritmu při kompresi/dekompresi, které jsou vhodné pro paralelizaci, implementovat a optimalizovat dané části v OpenCL a nakonec porovant rychlost implementace s CPU. Jelikož implementace JPEG kodéru a dekodéru je velmi pracnou záležitostí[1], rozhodli jsme se výkonnosti jednotlivých částí pipeline implementovat a měřit zvlášť.

Dále jsou v textu popsány vybrané algoritmy a porovnání paralelních variant na GPU se seriovými, které vykonává CPU.

2 Teoretický rozbor

Schéma pipeline pro JPEG kompresi na obrazku 1 obsahuje barevně odlišené bloky, což jsou algoritmy, které jsme se rozhodli paralelizovat. Mezi jednotlivými bloky jsou uvedeny i datové typy, jaké jsou mezi výstupem jednoho bloku a vstupem druhého bloku očekávány.

Převod barevného modelu

Diskrétní kosínova transformace slouží k převodu obrazu z prostorové domény do frekvenční domény. V normě JPEG je definováno 6 variant, z nichž nejčastěji používaná Baseline DCT, kterou jsme také v rámci projektu implementovali[2]. Po aplikaci kosínové transformace dostaneme dvě složky koeficientů – stejnosměrnou (AC) a střídavou (DC). Tyto složky se dále v algoritmu komprimují zvlášť. Vzorce 1, 2 a 3 jsou matematickým zápisem DCT, kde platí, že N je pro JPEG blok rovno 8.

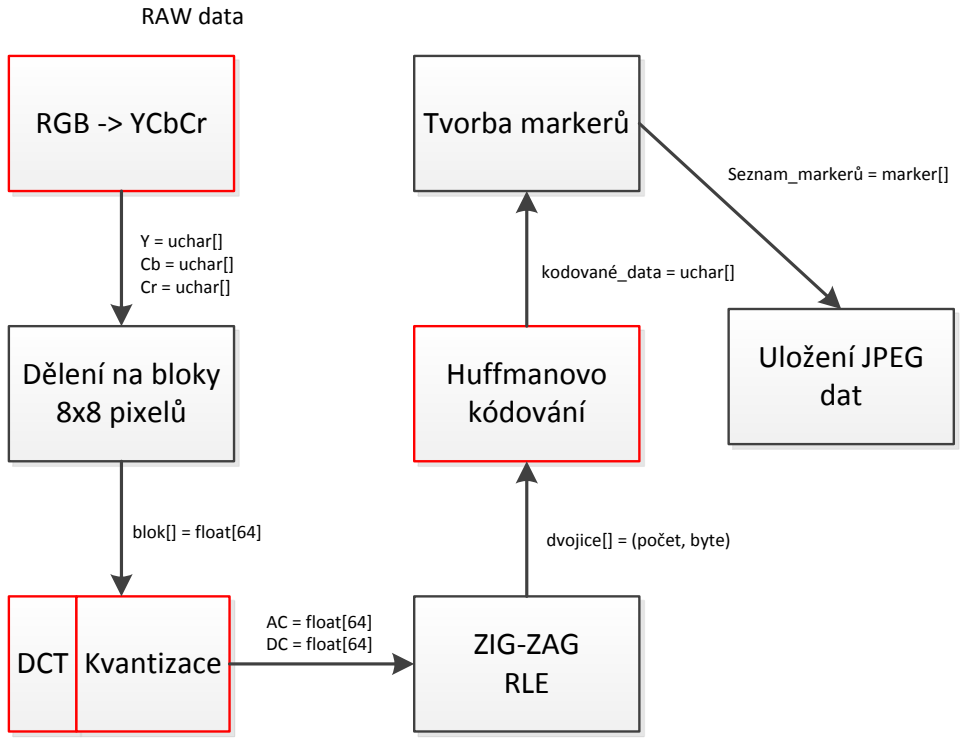
$$\lambda_x = \begin{cases} \frac{1}{\sqrt{2}}, & x = 0 \\ 1, & x \neq 0 \end{cases} \tag{1}$$

$$g_{k,j}[n,m] = \lambda_k \lambda_j \frac{2}{N} cos\left[\frac{k\pi}{N}\left(n+\frac{1}{2}\right)\right] cos\left[\frac{j\pi}{N}\left(m+\frac{1}{2}\right)\right] \tag{2}$$

$$c[k,j] = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} f[n,m] g_{k,j}[n,m] \tag{3}$$

Kvantizace je implementována pomocí dvou tabulek, zvlášť pro jasovou složku a chrominescenční složky. Hodnota v bloku je nejprve vydělena přislušnou hodnotou v tabulce, zaokrouhlena a poté vynásobena stejnou hodnotou z tabulky. Pro kvantizaci je definováno několik vhodných tabulek[1].

Huffmanovo kódování



Obrázek 1: Schéma JPEG encode pipeline.

3 Implementace

Celý projekt je rozdělen do několika modulů

- `ahed` –
- `cl_util` – funkce potřebné pro práci s OpenCL, jednotlivé kernely jsou v souboru `jpeg.cl`.
- `color_transform` –
- `dpcm_rle` –
- `huffman` –
- `jpeg_util` – tvorba a čtení JPEG markerů a další pomocné funkce pro JPEG, které nemají samostaný modul (zig-zag, kvantizace, DCT ...).
- `main` – funkce `main` a časová měření algoritmů.

Diskrétní kosínova transformace je implementována pomocí funkcí
`void dct8x8(float* block, float* dct, int* table)` – sériově
`void dct8x8_gpu(float* block, float* dst, cl_mem* table)` – paralélně
kde prvním argumentem jsou data bloku vstupujícího do DCT, druhým argumentem pole pro výstupní AC a DC koeficienty DCT a třetím argumentem je kvantizační tabulka, neboť v rámci optimalizace jsme sjednotili kvantizaci s DCT.

4 Závěr

Výsledkem naší práce je srovnání rychlosti implementací na těchto strojích.

- Ubuntu –
- Debian – Core i5 (2500K) 3.3 GHz, NVIDIA GeForce 8800 GTX
- OS X – Core i5 (I5-3317U) 1.7 GHz, HD Graphics 4000

Hodnoty měření jsou uvedeny v milisekundách, porovnávány jsou paralelní verze a sériové verze algoritmů.

Algoritmus	Ubuntu		Debian		OS X	
	serial [ms]	paralel [ms]	serial [ms]	paralel [ms]	serial [ms]	paralel [ms]
`RGB to YCbCr`	1	2	3	4	5	6
`YCbCr to RGB`	1	2	3	4	5	6
`Huffman`	1	2	3	4	5	6
`Inv_Huffman`	1	2	3	4	5	6
`DCT`	1	2	0.320911	0.144005	0.410795	0.264168
`Inv_DCT`	1	2	0.276089	0.113011	0.396967	0.203848

Tabulka 1: Tabulka srovnání doby výpočtu jednotlivých algoritmů.

Literatura

[1] *Recommendation T.81.* [cit. 2012-12-10]
<http://www.w3.org/Graphics/JPEG/itu-t81.pdf>

[2] D. Bařina. *Diskrétní kosínová transformace – prezentace ke cvičení.* [cit. 2008-12-08].
<http://www.fit.vutbr.cz/study/course-l.php.cs?id=8766>

[3] *Wikipedia, the free encyclopedia.* [cit. 2012-12-10]
<http://en.wikipedia.org/>