Architektura GPU a jejich využití pro obecné výpočty

Ing. Petr Pospíchal

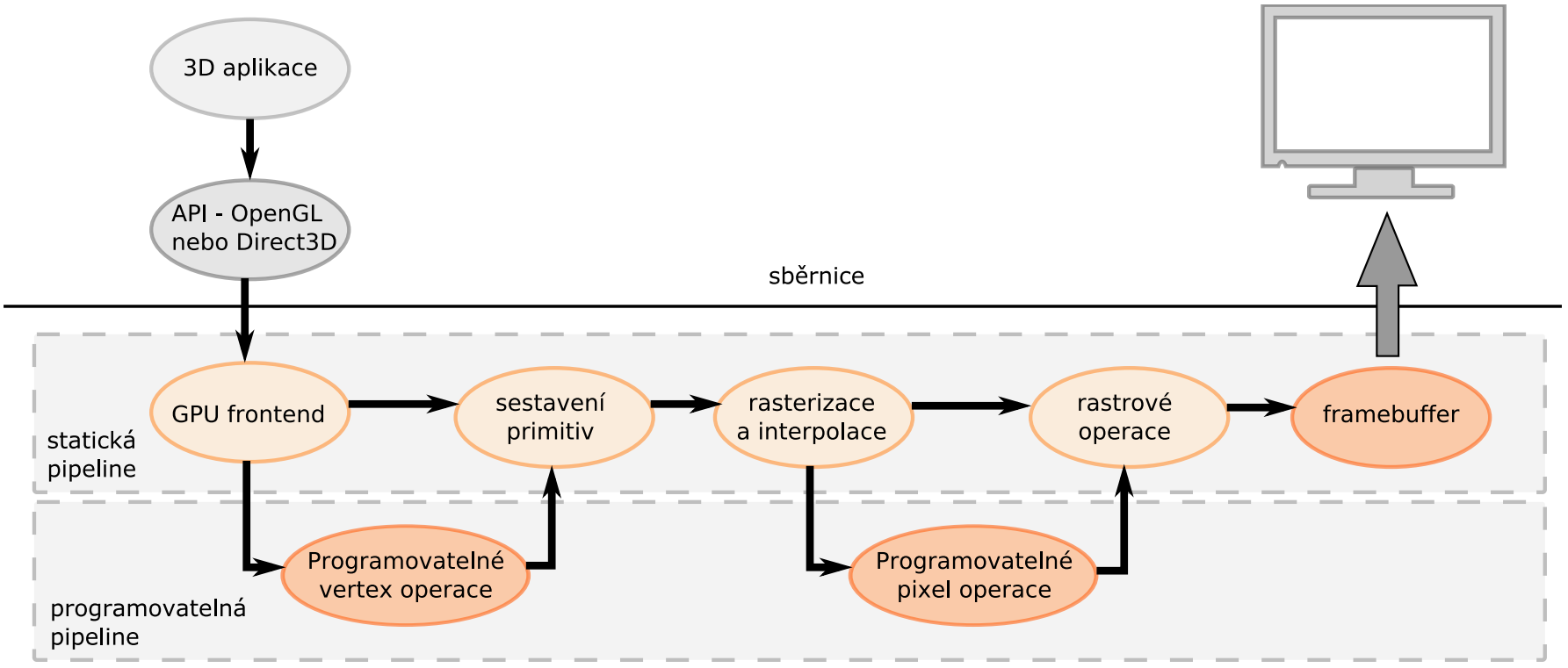
Ústav počítačových systémů Fakulty informačních technologií VUT v Brně

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Obsah prezentace

1. Historie GPGPU
2. Proč používat GPU?
3. CUDA
4. Nedostatky GPU příkladem

Grafická pipeline



Původně statická část, od 2001 s GeForce 3 i programovatelná.

Historie programovatelnosti GPU

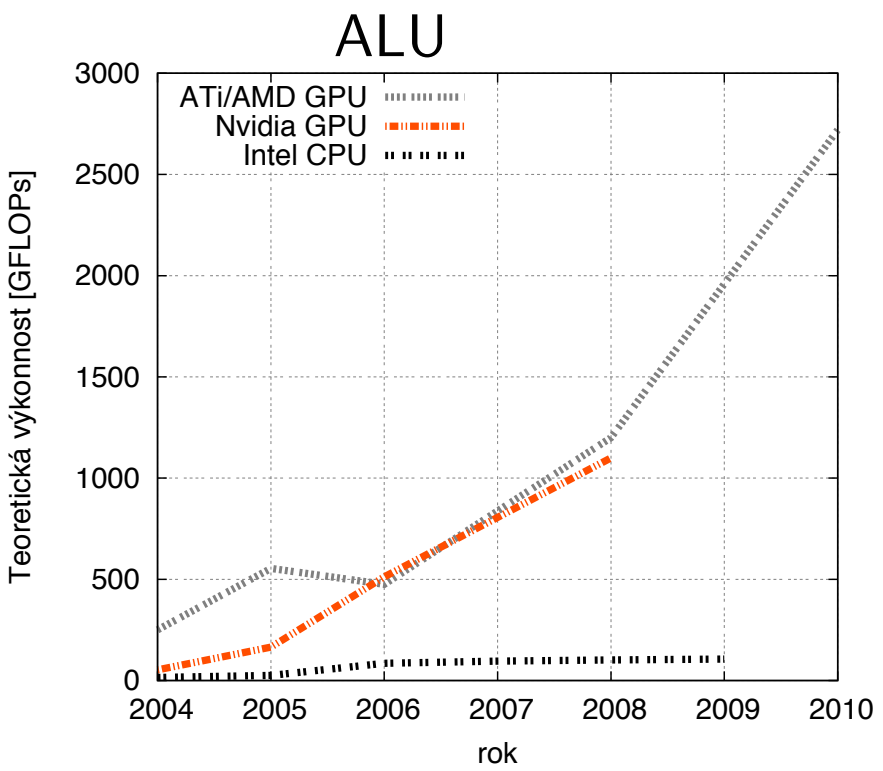
1999 Voodoo 3

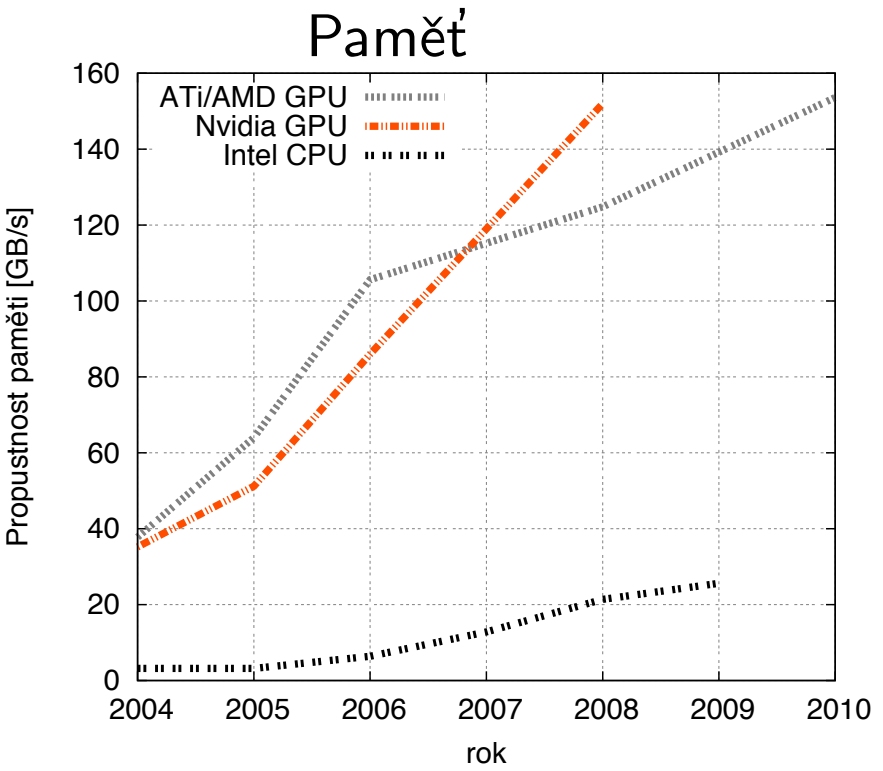
+obecnost výpočtů
+výkon

GeForce GTX 295 **2009**

	pixel+vertex jednotky				**unifikované jednotky**			
	Cg				CUDA	OpenCL	DirecX 11 compute	
rok	**2001**	**2002**	**2003**	**2004**	**2006**	**2007**	**2008**	**2009**
verze pixel a vertex programů	1.0	1.3	2.0	3.0	**4.0**	4.1		5.0
max. vykonaných instrukcí na jednotku	jednotky		stovky	tisíce	**bez limitu**			

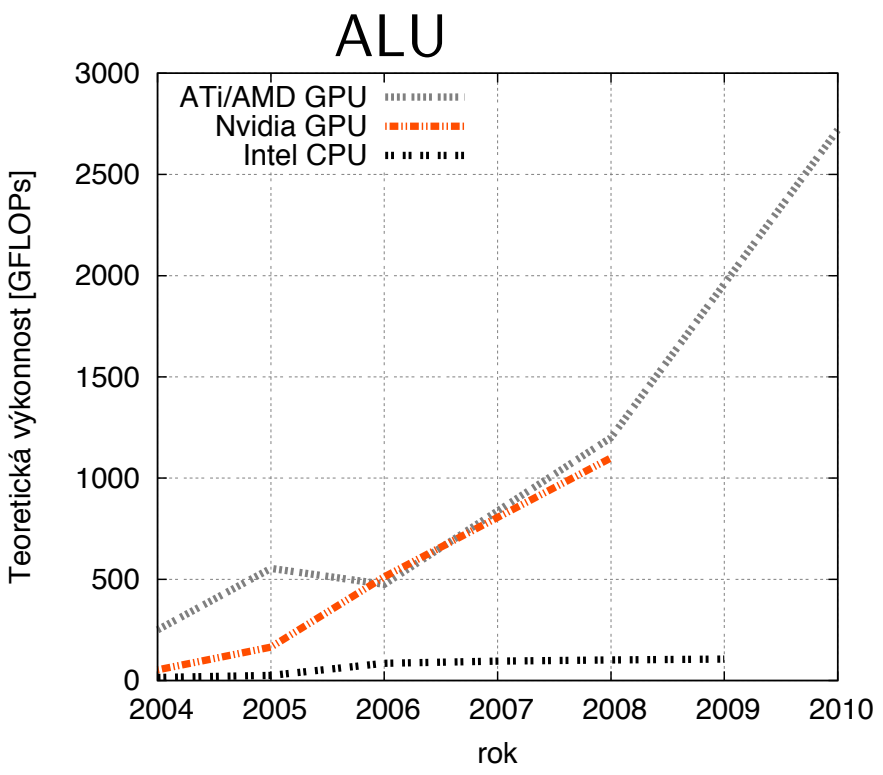
Grafické čipy vs procesory

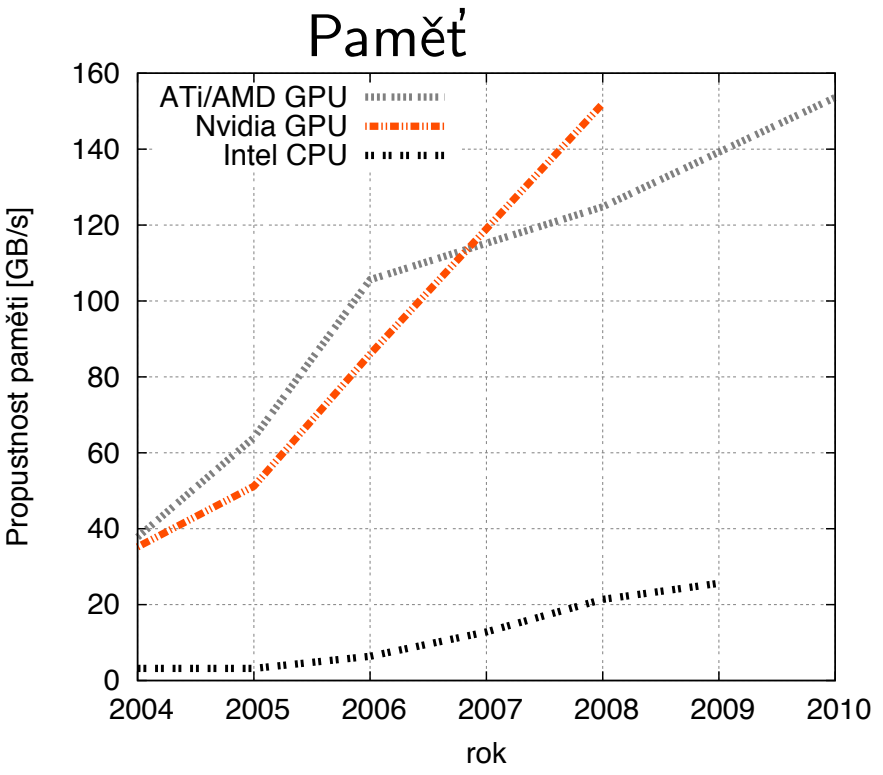




$\Rightarrow$ výkon ALU se rychle vzdaluje CPU, s pamětí to není tak slavné

Grafické čipy vs procesory





$\Rightarrow$ výkon ALU se rychle vzdaluje CPU, s pamětí to není tak slavné

Proč (ne)používat GPU?

výhody

- obrovský teoretický výkon
  GTX 280 - 240 jader = 1TFLOP, další generace 3TFLOP
- malá cena za jednotku výkonu
- malá spotřeba na jednotku výkonu
- hardwarová režie vláken
- rychlá paměť, sdílená paměť v rámci multiprocesoru
- externí adaptér, rozšiřitelnost

nevýhody

- špatné větvení
- omezené datové typy, double je mnohem pomalejší
- nízký výkon na vlákno, potřeba masivně paralelních výpočtů
- úzké hrdlo na sběrnici

Proč (ne)používat GPU?

výhody

- obrovský teoretický výkon
  GTX 280 - 240 jader = 1TFLOP, další generace 3TFLOP
- malá cena za jednotku výkonu
- malá spotřeba na jednotku výkonu
- hardwarová režie vláken
- rychlá paměť, sdílená paměť v rámci multiprocesoru
- externí adaptér, rozšiřitelnost

nevýhody

- špatné větvení
- omezené datové typy, double je mnohem pomalejší
- nízký výkon na vlákno, potřeba masivně paralelních výpočtů
- úzké hrdlo na sběrnici

Porovnání dvou generací DX10 GPU: cesta paralelizace

karta (GPU)	parametr	hodnota
GeForce 8800 GTX (G80)	velikost hlavní paměti	768MB
	propustnost hlavní paměti	86.4 GBps
	takt jádra	575 Mhz
	takt shader jednotek	1.35 Ghz
	max. příkon	155 W
	hrubý výkon	518 GFLOPs
	shader jednotek	128 (8x16)
GeForce GTX 285 (GT200)	velikost hlavní paměti	1024MB
	propustnost hlavní paměti	159 GBps
	takt jádra	648 Mhz
	takt shader jednotek	1.47 Ghz
	max. příkon	183 W
	hrubý výkon	1062 GFLOPs
	shader jednotek	240 (8x30)

Compute Unified Device Architecture

- framework firmy nVidia pro obecné výpočty na GPU (GPGPU)
- funguje na GeForce 8 a výš pod Windows a Linux (DirectX 10 generace s unified shadery)
- úzká vazba na hardware, možnost využití rychlé sdílené paměti
- vychází z jazyka C s přidanými direktivy pro vyjádření paralelismu
- slouží jako základ pro OpenCL

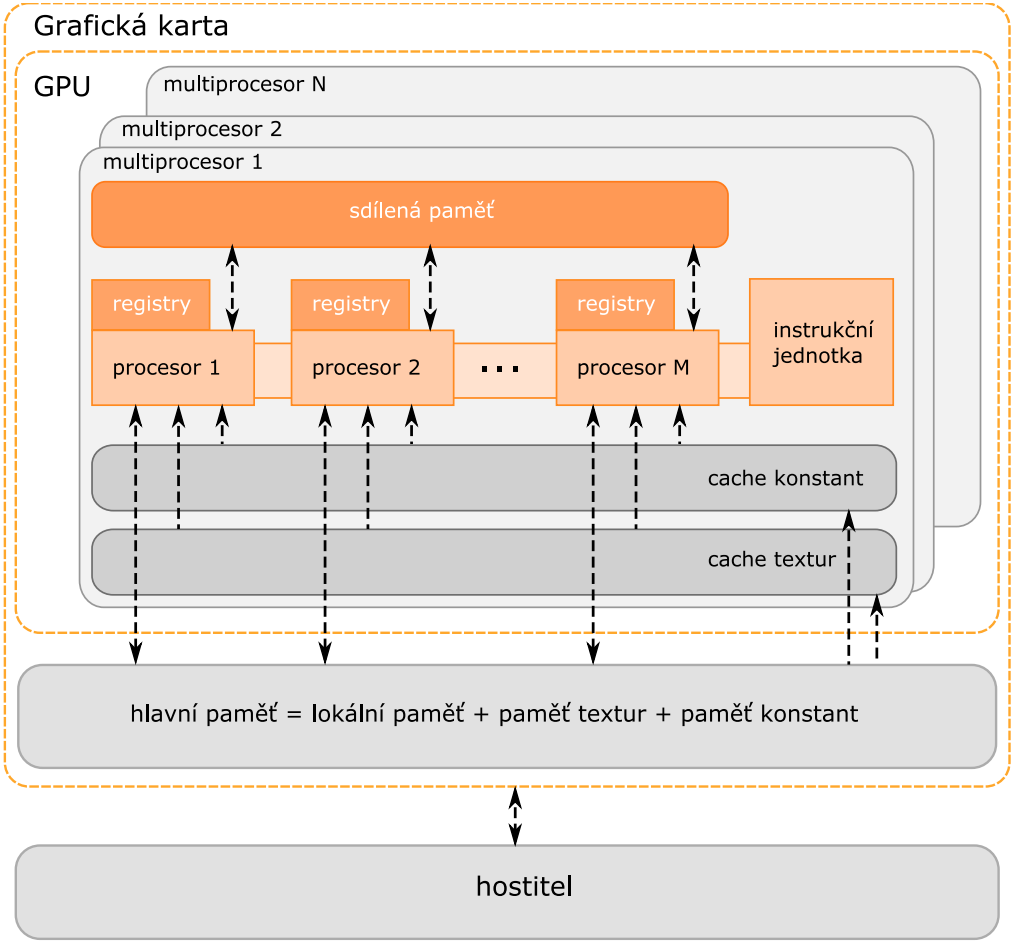
$\Rightarrow$ výzva je vytvořit program optimalizovaný na GPU HW

Compute Unified Device Architecture

- framework firmy nVidia pro obecné výpočty na GPU (GPGPU)
- funguje na GeForce 8 a výš pod Windows a Linux (DirectX 10 generace s unified shadery)
- úzká vazba na hardware, možnost využití rychlé sdílené paměti
- vychází z jazyka C s přidanými direktivy pro vyjádření paralelismu
- slouží jako základ pro OpenCL

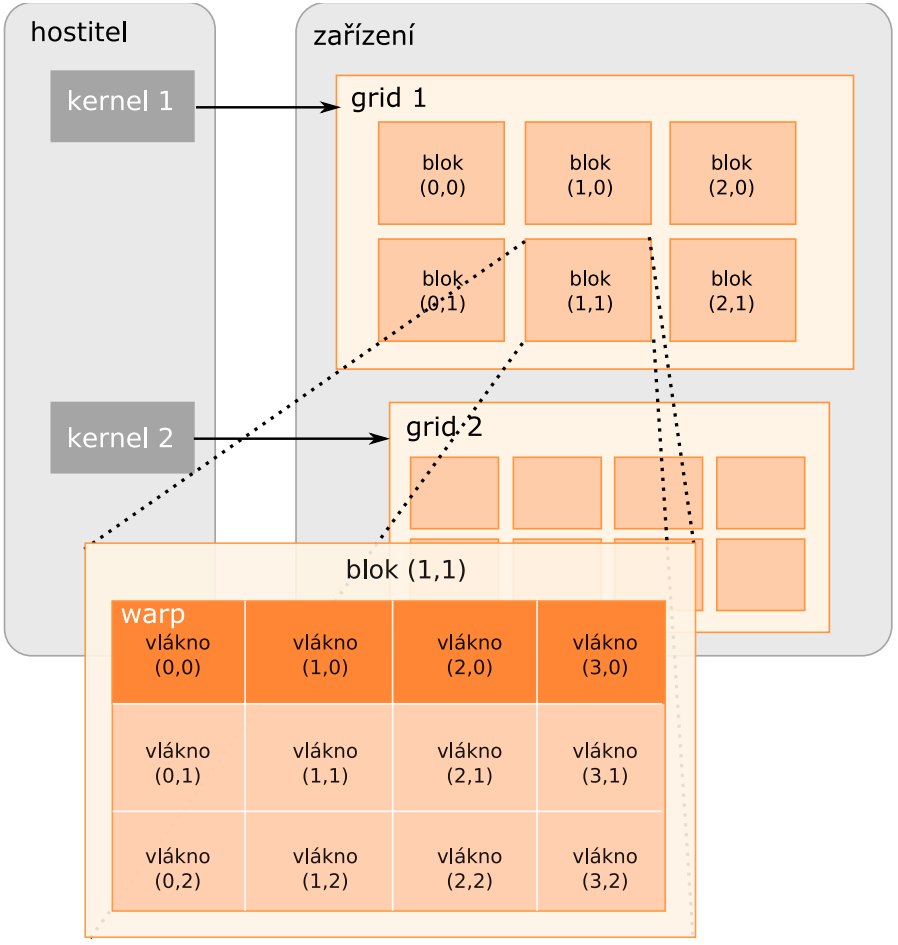
$\Rightarrow$ **výzva je vytvořit program optimalizovaný na GPU HW**

Hardwarový model



- grafická karta = GPU + hlavní paměť
- hostitel komunikuje s grafickou kartou skrz PCI Express 16x zápisem do hlavní paměti
- hardwarový scheduler vláken
- SIMD model: jediná instrukční jednotka na multiprocesor
- skrývání latence paměti přepínáním vláken
- instrukce optimalizované na zpracování obrazu

Softwarový model



- dělení výpočtu do nezávislých bloků skládajících se z vláken
- možnost snadné synchronizce vláken v bloku, ale ne bloků mezi sebou
- bloky jsou mapovány na multiprocesory
- warp vláken běží fyzicky paralelně

Hierarchie paměti

- cache textur je optimalizovaná na 2D lokalitu
- sdílená paměť má 16KB na multiprocesor
- pameť mimo čip je ve všech případech hlavní a má kapacitu stovky MB. U textur a konstant je cachována

Tabulka: Porovnání paměťové hierarchie

druh	přístup	umístění	operace	cache
Registry	Jedno vlákno	Na čipu	RW	Ne
Sdílená	Vlákna uvnitř bloku	Na čipu	RW	Ne
Lokální	Jedno vlákno	Mimo čip	RW	Ne
Globální	Vlákna + host	Mimo čip	RW	Ne
Texturová	Vlákna + host	Mimo čip	vlákna RO	6-8KB/multi.
Konstantní	Vlákna + host	Mimo čip	vlákna RO	8KB/multi.

Efektivní využití hlavní paměti

vlastnosti

- vysoká latence 400-600 taktů
- rychlost je kompenzována šířkou sběrnice

důsledky

- pro efektivní využití je třeba sdruženého přístupu half-warpu
- v tom případě jsou všechna data half-warpu načtena jako blok jednou transakcí
- pro sdružený přístup je potřeba splnit následující podmínky:
  1. Velikost paměťového elementu, ke které přistupujeme, je 4, 8 nebo 16 bytů (např. int, float, ale už ne char!).
  2. Vlákna k elementům přistupují sekvenčně: tedy k n-tému elementu pouze n-té vlákno.
  3. Všech 16 elementů leží ve stejném segmentu, přičemž adresa prvního elementu musí být zarovnána k 16násobku velikosti elementu.

Efektivní využití hlavní paměti

vlastnosti

- vysoká latence 400-600 taktů
- rychlost je kompenzována šířkou sběrnice

důsledky

- pro efektivní využití je třeba sdruženého přístupu half-warpu
- v tom případě jsou všechna data half-warpu načtena jako blok jednou transakcí
- pro sdružený přístup je potřeba splnit následující podmínky:
  1. Velikost paměťového elementu, ke které přistupujeme, je 4, 8 nebo 16 bytů (např. int, float, ale už ne char!).
  2. Vlákna k elementům přistupují sekvenčně: tedy k n-tému elementu pouze n-té vlákno.
  3. Všech 16 elementů leží ve stejném segmentu, přičemž adresa prvního elementu musí být zarovnána k 16násobku velikosti elementu.

Efektivní využití sdílené paměti

vlastnosti

- rychlost je na úrovni registrů (2 takty), avšak kapacita je silně omezená (16KB na multiprocesor)
- dělení do banků o velikosti 4 byte

důsledky

- při paralelním přístupu více vláken do jednoho banku musí být přístup serializován (N-way bank conflict)
- jediná výjimka neserializovaného vícenásobného přístupu je broadcast
- rozestup mezi daty vláken by měl být 4 byte
- při správném použití VELMI dobrý pomocník

Efektivní využití sdílené paměti

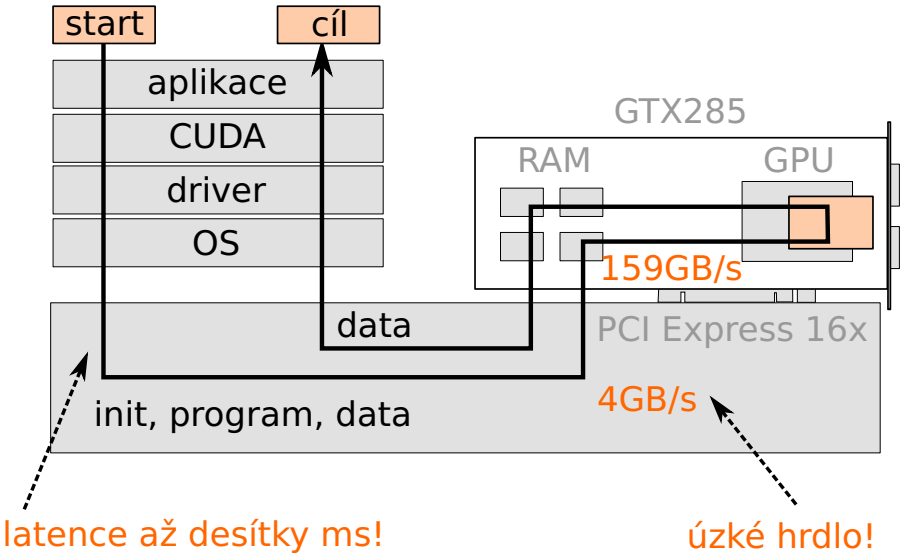
vlastnosti

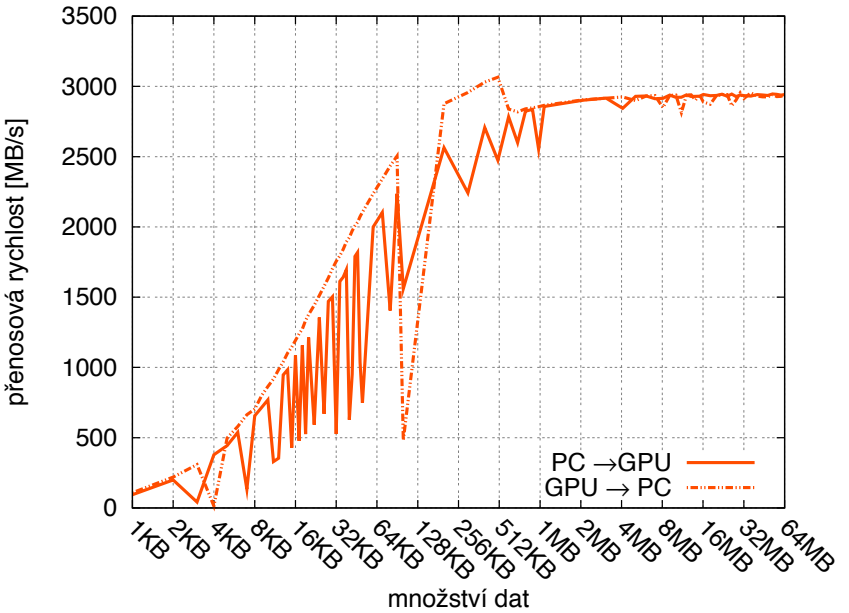
- rychlost je na úrovni registrů (2 takty), avšak kapacita je silně omezená (16KB na multiprocesor)
- dělení do banků o velikosti 4 byte

důsledky

- při paralelním přístupu více vláken do jednoho banku musí být přístup serializován (N-way bank conflict)
- jediná výjimka neserializovaného vícenásobného přístupu je broadcast
- rozestup mezi daty vláken by měl být 4 byte
- při správném použití VELMI dobrý pomocník

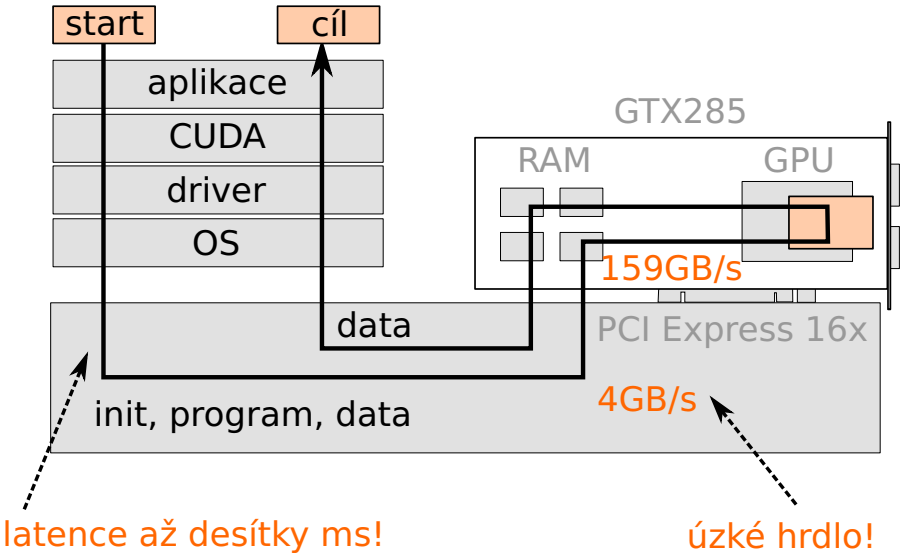
GPGPU: Průběh výpočtu a přenosy na sběrnici

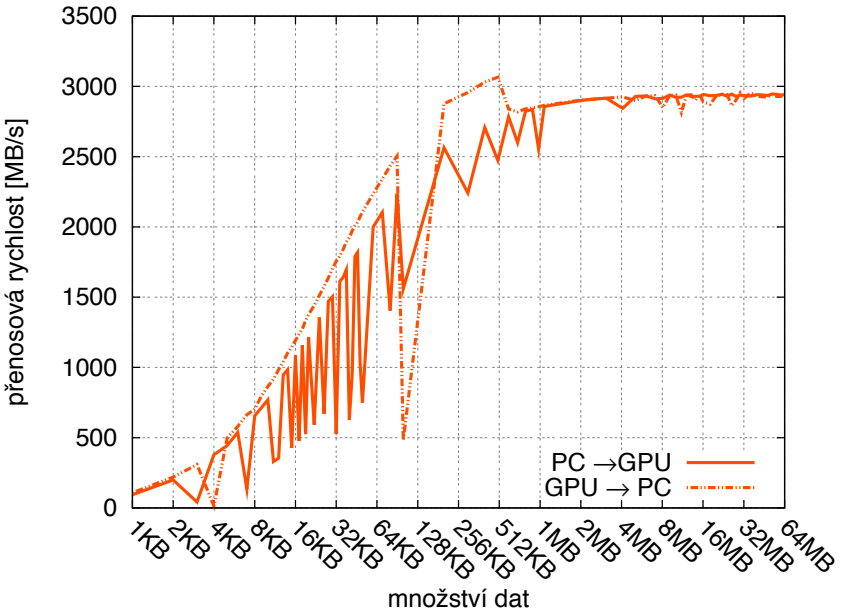




⇒ pokud možno celý výpočet na GPU

GPGPU: Průběh výpočtu a přenosy na sběrnici



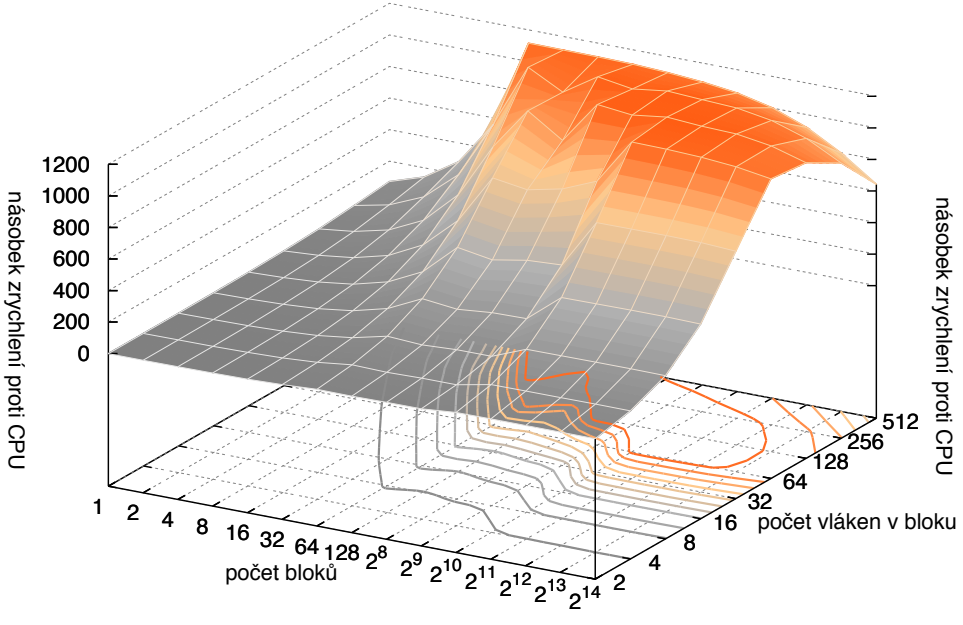


⇒ **pokud možno celý výpočet na GPU**

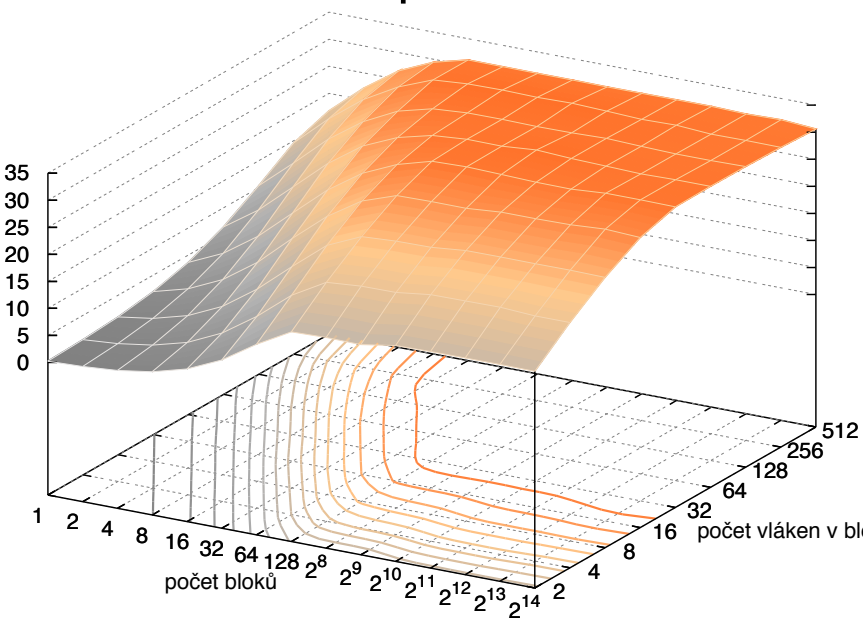
Vliv sběrnice a počtu vláken na zrychlení

8 milionů vyhodnocení Michalewiczovy funkce

běh na GPU



běh na GPU + přenos dat

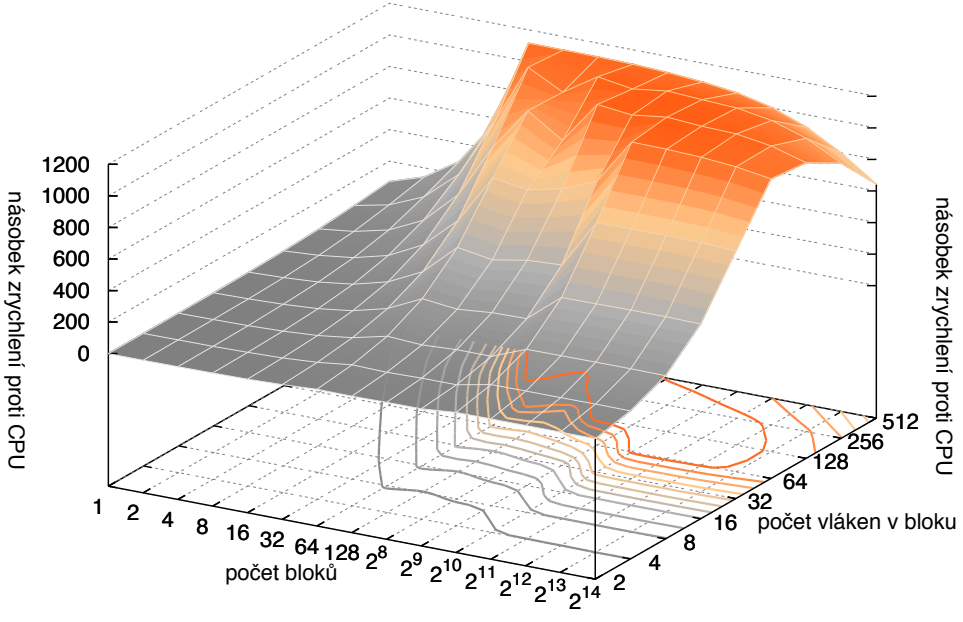


$\Rightarrow$ minimalizovat komunikaci, používat asynchronní přenosy, je potřeba tisíců vláken

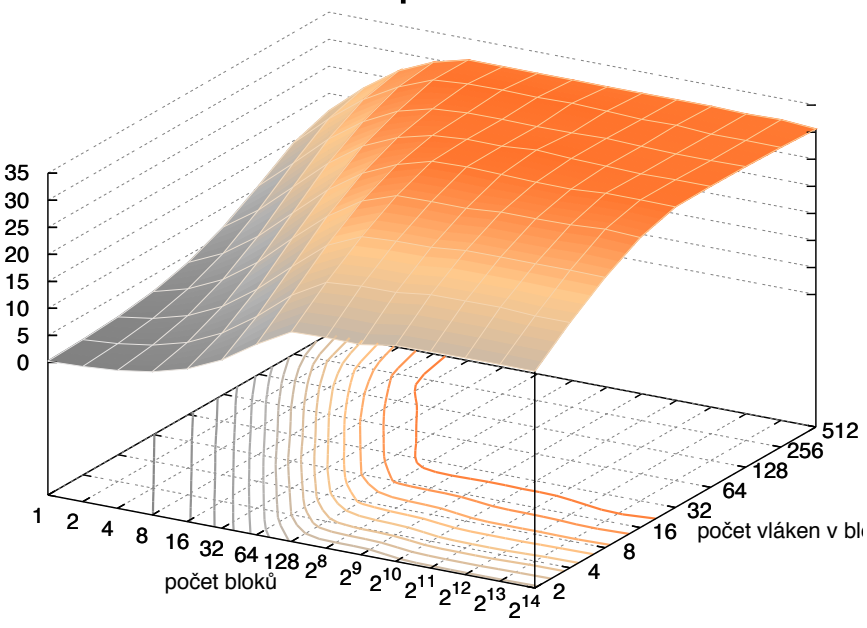
Vliv sběrnice a počtu vláken na zrychlení

8 milionů vyhodnocení Michalewiczovy funkce

běh na GPU

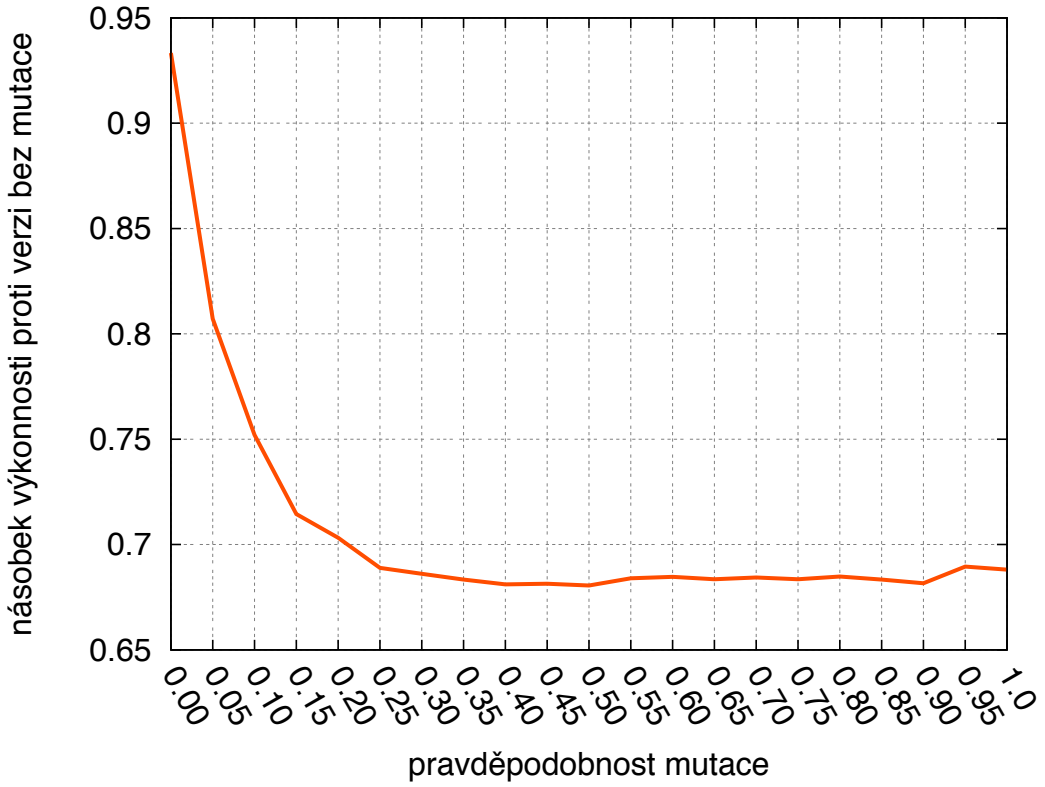


běh na GPU + přenos dat



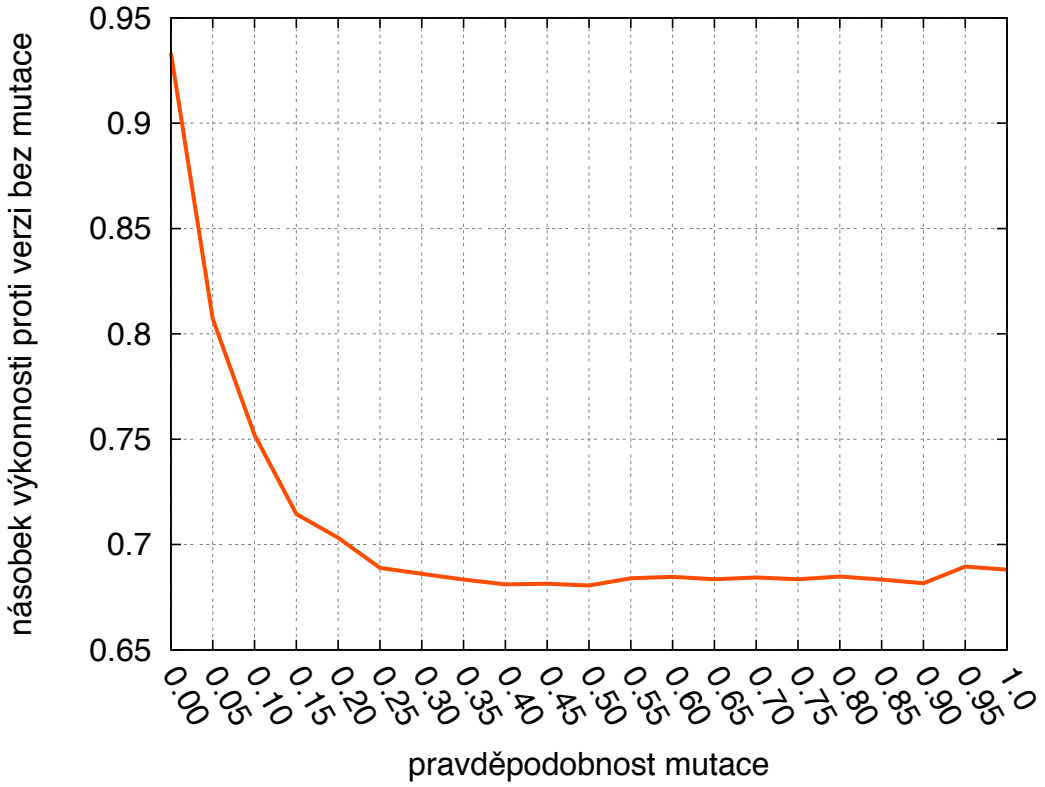
⇒ minimalizovat komunikaci, používat asynchronní přenosy, je potřeba tisíců vláken

Vliv divergence vláken na rychlost



$\Rightarrow$ nutno navrhovat SIMD-přátelské aplikace

Vliv divergence vláken na rychlost



⇒ **nutno navrhovat SIMD-přátelské aplikace**

Závěr

Děkuji za pozornost

Nyní rád zodpovím vaše dotazy!

Závěr

Děkuji za pozornost

Nyní rád zodpovím vaše dotazy!