Paralelizace genetických algoritmů pomocí GPGPU

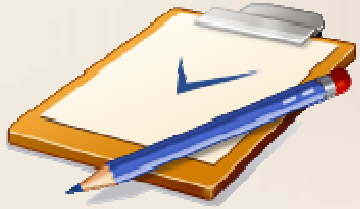
Diplomová práce z SI

Bc. Jan Hofta

2008 - 2009

Osnova práce

- Cíle práce:
  - Nastudovat dostupné zdroje o paralelních GA
  - Vybrat verzi GA vhodnou pro paralelizaci
  - Provést paralelizaci GA na grafickém procesoru
- Části práce:
  - 1. kapitola – Genetické algoritmy
  - 2. kapitola – NVIDIA CUDA
  - 3. kapitola – Implementace
  - 4. kapitola – Výsledky



Genetické algoritmy (1)

- Vychází z evoluční teorie
- Nástroj pro hledání optima funkce
- Idea algoritmu:
  - 1. body z definičního oboru funkce
  - 2. hodnotu binárně zakódovat (tzv. chromosomy), dohromady 0. generace populace
  - 3. ohodnotit kvalitu řešení (tzv. fitness funkcí)

Genetické algoritmy (2)

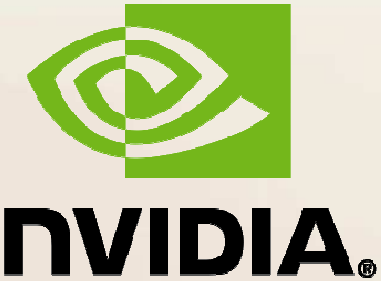
- 4. nalézt partnery (tzv. selekce)
- 5. vyměnit část řetězce mezi partnery (tzv.křížení)
- 6. na výsledcích náhodné změny (tzv. mutace)
- 7. ohodnotit nové, nahradit staré
- 8. tím další generace a opět bod 4
- 9. po nějaké době stop, nejkvalitnější chromosom = argument extrému funkce

Paralelní GA

- Farmářský model
  - Jedna populace, hlavní a pomocné procesy
- Migrační model
  - Více populací, migrace
- Difuzní model
  - 1 proces = 1 jedinec
- Hybridní model
  - Kombinace ostatních ve více úrovních

NVIDIA CUDA

- Společnost NVIDIA, od roku 2007
- Snaha přenést paralelní výpočty na grafickou kartu
- Mnoho paralelních procesů (tzv. vlákna), spojena do tzv. bloků a ty dále do tzv. gridů
- Velice přirozené, rozšíření jazyka C a knihovna funkcí pro procesor (hostitel) a grafickou kartu (zařízení)

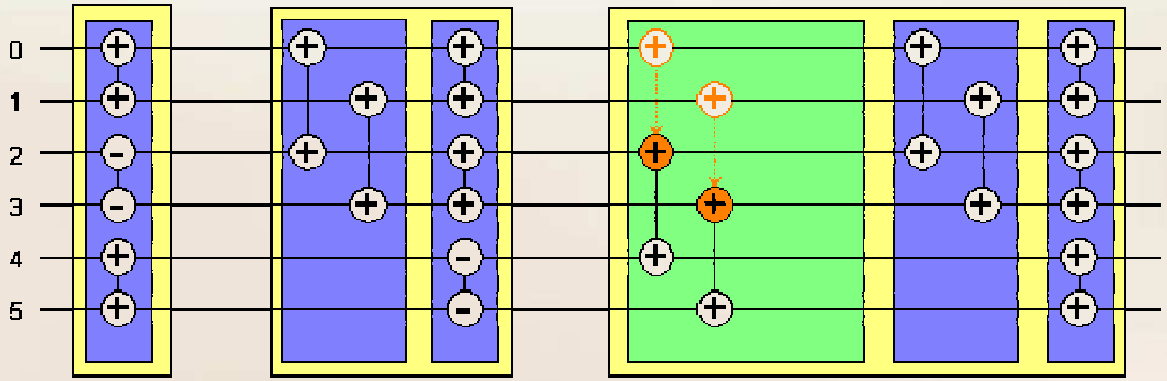


Implementace

- Naprogramovány tři algoritmy
  - Základní sekvenční GA (GASimple)
  - Difuzní model paralelního GA prováděný na procesoru s bitonickým tříděním na GPU (CPUPGA)
  - Difuzní model paralelního GA prováděný kompletně na grafické kartě (PGA)

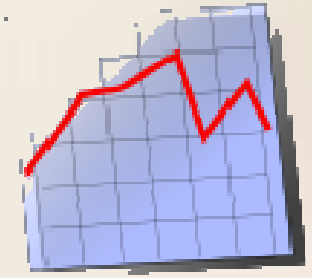
Implementační výzvy

- Bitonické třídění (paralelní třídící algoritmus)
  - K řešení:
    - Vhodná implementace pro daný problém
    - Zobecnění pro poslopnosti jiných délek než 2^p
- Synchronizace dat mezi bloky na GPU
- Implementace náhody



Výsledky

- Porovnání výpočetní doby a kvality řešení v závislosti na počtu generací a jedinců
- Testované funkce:
  - Pět funkcí dvou proměnných
  - Tři funkce čtyř proměnných
  - Jedna funkce deseti proměnných
- Konfigurace:
  - 1000 generací, 225 jedinců
  - 2000 generací, 225 jedinců
  - 1000 generací, 441 jedinců
  - 2000 generací, 441 jedinců



Testované funkce

- 2 proměnné:

$$f_1(x,y) = -x.\sin\sqrt{|x-(y+47)|} - (y+47).\sin\sqrt{\left|y+47+\frac{x}{2}\right|}$$

$$f_2(x,y) = 0.5 + \frac{\sin^2\sqrt{x^2(100+y^2)} - 0.5}{(1+0.001(x-y)^2)^2}$$

$$f_3(x,y) = -20e^{-0.02\sqrt{\frac{x^2+y^2}{2}}} - e^{\frac{\cos 2\pi x + \cos 2\pi y}{2}} + 20 + e$$

$$f_4(x,y) = 100(x^2-y)^2 + (1-x)^2$$

$$f_5(x,y) = 1 - \cos x \cos y + \frac{x^2+y^2}{4000}$$

- 4 proměnné:

$$f_6(x_1,x_2,x_3,x_4) = \sum_{i=1}^{3} 100(x_{i+1}-x_i^2)^2 + (1-x_i)^2$$

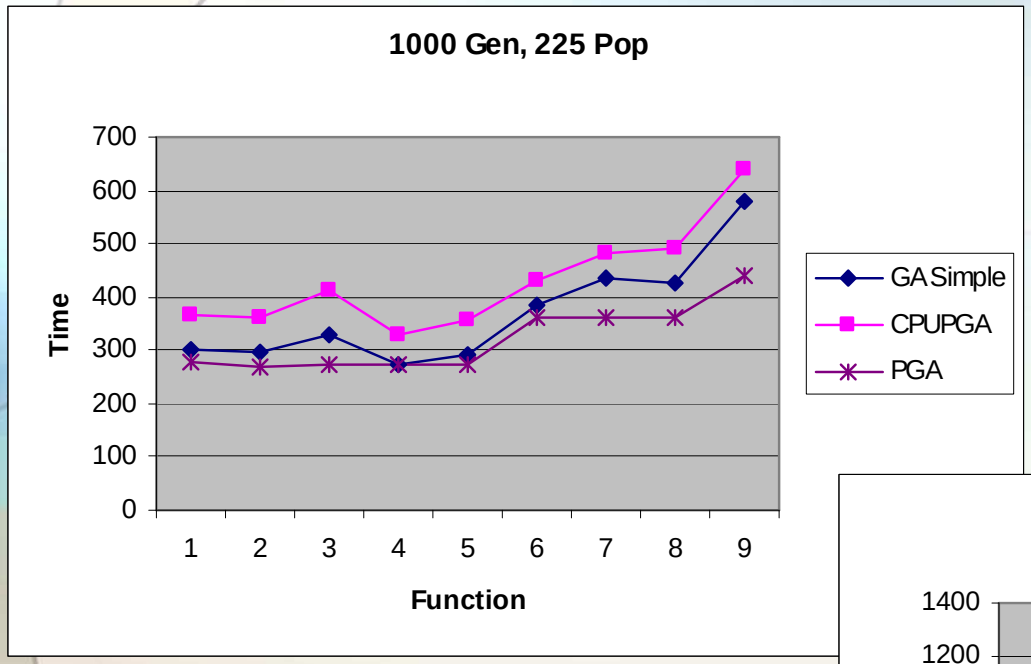
$$f_7(x_1,x_2,x_3,x_4) = \sum_{i=1}^{4} -x_i \sin\sqrt{|x_i|}$$

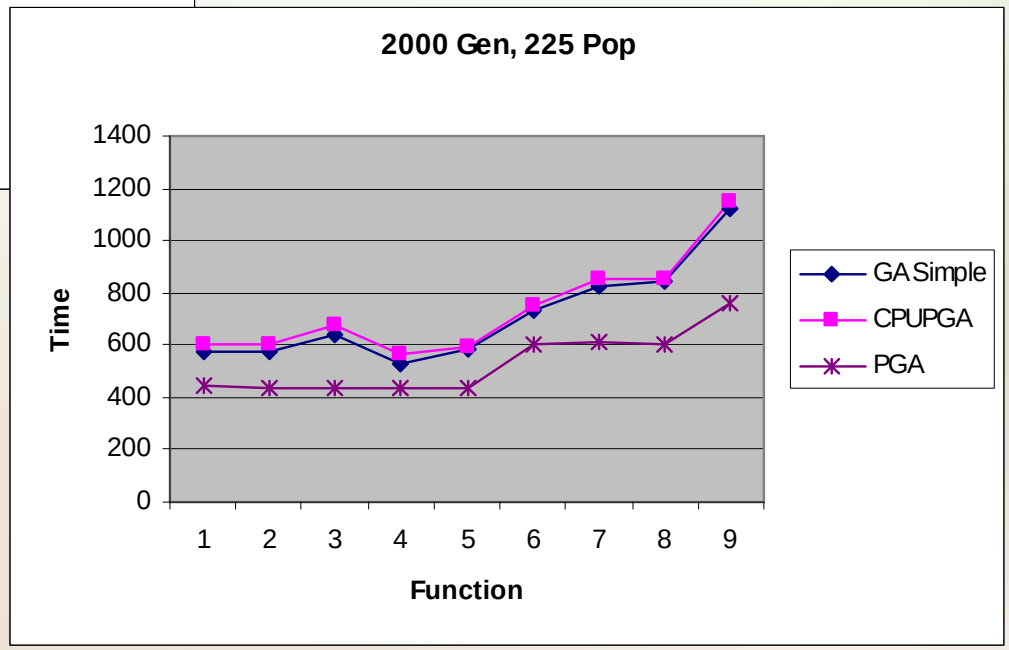
$$f_8(x_1,x_2,x_3,x_4) = \sum_{i=1}^{4} \frac{x_i^2}{4000} - \prod_{i=1}^{4} \cos x_i + 1$$

- 10 proměnných:

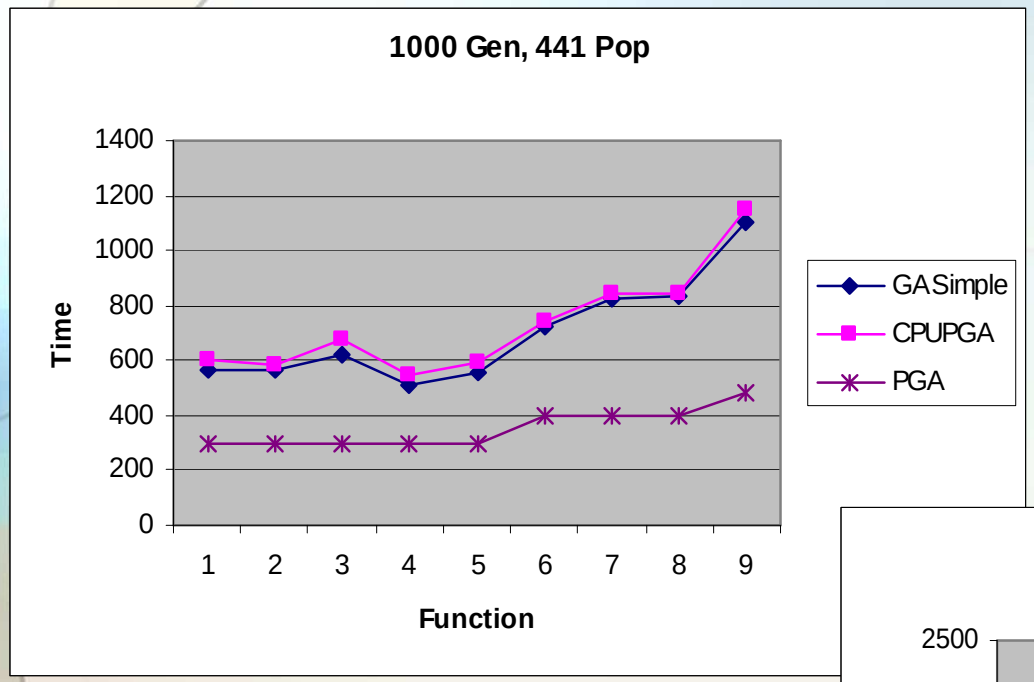
$$f_{7,10\dim}(x_1,x_2,\ldots,x_{10}) = \sum_{i=1}^{10} -x_i \sin\sqrt{|x_i|}$$

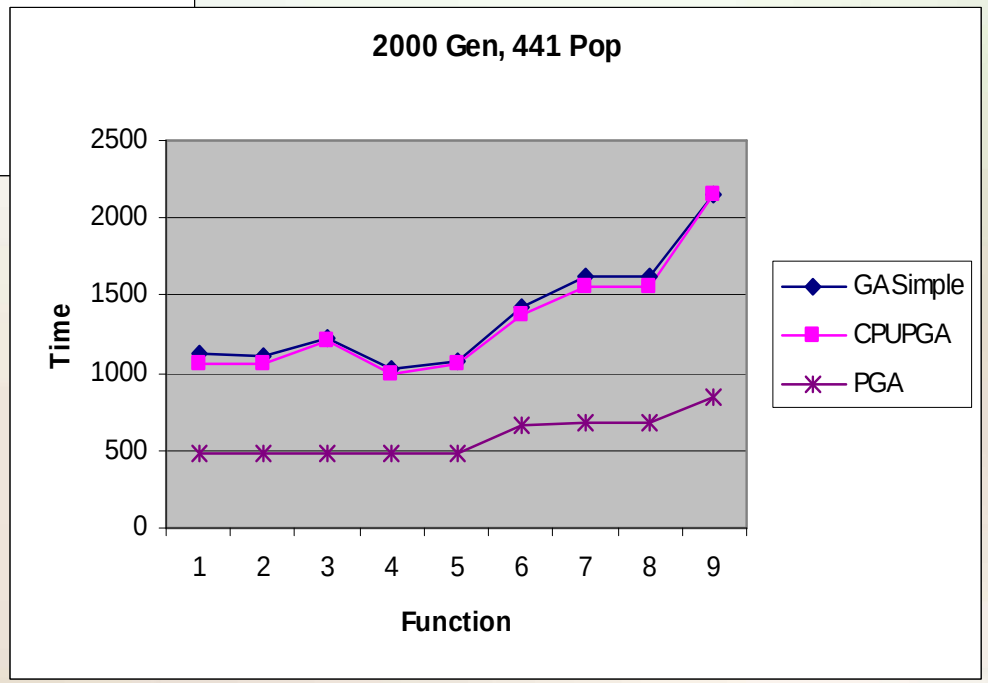
Časová náročnost (1)





Časová náročnost (2)





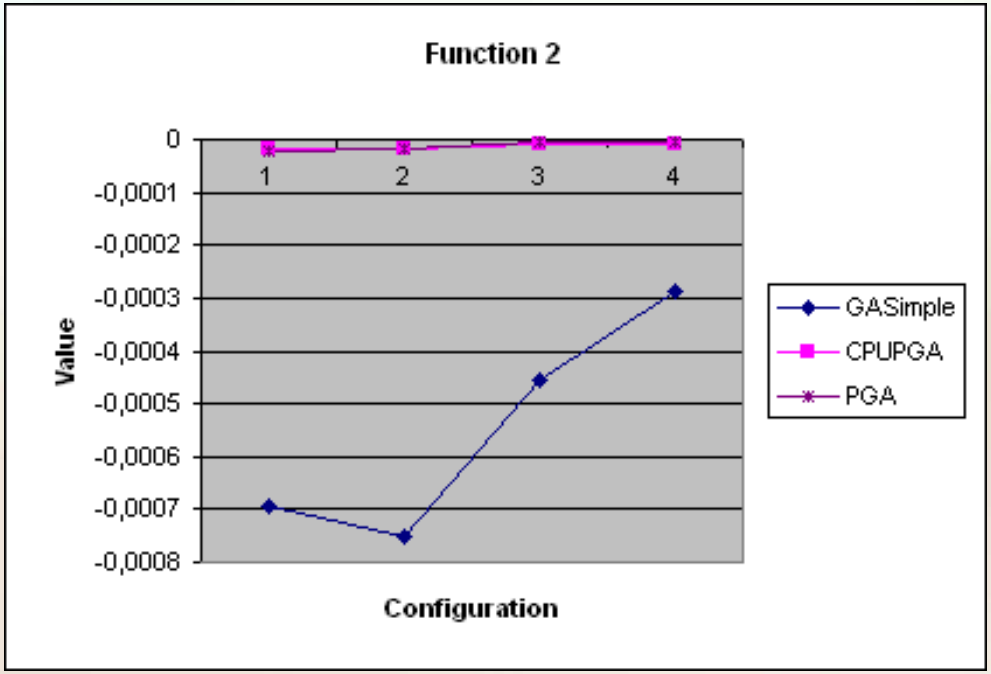
Časová náročnost (3)

Konfigurace	2 proměnné		4 proměnné		10 proměnných	
	GASimple	CPUPGA	GASimple	CPUPGA	GASimple	CPUPGA
1000 g, 225 p	1,09	1,33	1,15	1,29	1,31	1,45
2000 g, 225 p	1,33	1,38	1,32	1,35	1,47	1,50
1000 g, 441 p	1,91	2,03	2,01	2,05	2,29	2,40
2000 g, 441 p	2,33	2,25	2,31	2,22	2,58	2,58

Kvalita řešení

- Obecně: difuzní model = kvalitnější řešení
- Příklad:

$$f_2(x,y) = 0.5 + \frac{\sin^2\sqrt{x^2(100+y^2)} - 0.5}{(1+0.001(x-y)^2)^2}$$



C1 – 1000 Gen, 225 Pop

C2 – 2000 Gen, 225 Pop

C3 – 1000 Gen, 441 Pop

C4 – 2000 Gen, 441 Pop

Závěry a přínos práce

- Závěry:
  - Časová výhodnost použití PGA - roste s počtem generací a výrazně s počtem jedinců
  - Kvalitnější řešení od difuzního modelu
- Přínos:
  - Nalezena vhodná verze GA pro paralelizaci na GPU
  - Provedena a vyzkoušena implementace pomocí NVIDIA CUDA
  - Zobecnění a implementace bitonického třídění

Konec

- Děkuji Vám za pozornost