# Architektura procesorů (ACH 2016)
## Projekt č. 2: CUDA

Filip Vaverka (ivaverka@fit.vutbr.cz)

Termín odevzdání: 18. 12. 2016

## 1 ÚVOD

V prvním projektu jste si mohli vyzkoušet optimalizaci sekvenčního kódu pomocí vektorizace na jednoduchém příkladu násobení matice a vektoru a na problému částicového systému. Cílem tohoto projektu bude implementovat částicový systém na grafické kartě pomocí technologie CUDA. Veškerý kód bude spouštěn na superpočítači Anselm.

## 2 CUDA NA SUPERPOČÍTAČI ANSELM

Pro připojení na superpočítač Anselm postupujte dle návodu v prvním projektu. Pouze při vytváření úlohy zvolte frontu `qnvidia`:

```
[ivaverka@login1.anselm ~]$ qsub -A DD-16-43 -q qnvidia -l select=1:ncpus=16,walltime=1:00:00 -I
qsub: waiting for job 1307408.dm2 to start
qsub: job 1307408.dm2 ready

[ivaverka@cn182.anselm ~]$
```

Následně je nutné načíst moduly `intel` a `cuda`:

```
module load intel/15.3.187
module load cuda
```

# 3 Částicový systém (15 bodů)

Při implementaci částicového systému vyjděte z teorie uvedené v zadání prvního projektu a využijte své dosažené výsledky, zaměřte se na správný výpočet gravitační síly!

## 3.1 Krok 0: základní implementace (7 bodů)

Kostra aplikace je připravena v adresáři `step0`. Nejprve správně doplňte definici struktury `t_particles` v hlavičkovém souboru `nbody.h`. Použijte vhodné datové typy tak, aby se omezil počet přístupů do globální paměti. Doplňte také funkce `particles_read` a `particles_write` pro načítání a ukládání dat.
Dalším krokem bude doplnění vyznačených míst v souboru `main.cu` – je třeba doplnit alokaci paměti na CPU a GPU, kopírování načtených dat z CPU do GPU a zpět a spouštění kernelu na GPU. Na GPU alokujte vše dvakrát, v každém kroku výpočtu pak použijte jednu kopii dat jako vstupy (`p_in`) a druhou jako výstupy (`p_out`). Tím odpadne nutnost synchronizace vláken před zápisem do paměti. V každém dalším kroku pak tyto dvě kopie vždy prohoďte.
Následně implementujte samotný kernel `particles_simulate` v souboru `nbody.cu` tak, aby kernel správně simuloval pohyb jedné částice s časovým posuvem `dt` sekund. V souboru `Makefile` nastavte počet vláken na blok a tuto proměnnou thr_blc společně s počtem částic `N` pak použijte při spouštění kernelu. Program přeložíte příkazem `make` a spustíte pomocí `make run`.
Správnost výpočtu je možné ověřit porovnáním výstupního souboru se vzorovým výstupem `output.dat`. Odchylky v řádech desetin značí, že je ve výpočtu významná chyba. Řádově menší chyby mohou být způsobeny i mírně odlišným výpočtem, dokonce i přeuspořádáním operací, a nebudou hodnoceny negativně. Po ověření správnosti výpočtu najděte ideální nastavení `THREADS_PER_BLOCK` tak, aby byl výpočet co nejrychlejší. Do souboru `nbody.txt` zapište výsledné časy.
Pro ladění výkonnosti použijte profilování, pomocí příkazu `make profile` spusťte profilovací nástroj `nvprof` s předpřipravenými metrikami. Seznam všech dostupných metrik získáte příkazem `nvprof -query-metrics`. Analyzujte přichystané i Vámi přidané metriky a na jejich základě optimalizujte svůj kód.

## 3.2 Krok 1: sdílená paměť (5 bodů)

Zkopírujte celý adresář `step0` do nového adresáře `step1`. V tomto kroku využijte sdílené paměti, abyste omezili přístupy do globální paměti. Funkčnost řešení ověřte srovnáním výstupů simulací! Porovnejte výkonnost s předchozím krokem. Dochází ke zrychlení? Zdůvodněte.
Pomocí profilování zjistěte rozdíly mezi implementacemi v kroku 1 a 2 a tyto rozdíly popište v souboru `nbody.txt`.

## 3.3 Krok 2: analýza výkonu (3 body)

Pomocí programu `gen` generujte datové soubory různých velikostí (volte mocniny dvou). Např. pro vygenerování souboru s 4096 částicemi použijte následující příkaz:

```
./gen 4096 4096.dat
```

Pro každý počet částic stanovte ideální počet vláken na blok a zapište výsledný čas do souboru `nbody.txt`. Naměřené časy porovnejte se sekvenční implementací z prvního projektu a spočtěte zrychlení. Od jakého počtu částic se vyplatí použít grafickou kartu (uvažujte, že paralelní verze na CPU bude cca 10× rychlejší než sekvenční verze)?

### 3.4 Bonus: jiná mikroarchitektura GPU (až 2 body)

Pokud budete mít zájem o bonusové body, změřte výkonnost obou implementací (krok 0 i 1) na grafické kartě s jinou mikroarchitekturou, např. Tesla (pozor, neplést označení produktové řady GPGPU[1] a označení mikroarchitektury[2]. Porovnejte rozdíly v mikroarchitektuře oproti kartám na Salomonu a vysvětlete, jaký to má dopad na výkonnost jednotlivých implementací.

## 4 Výstup projektu a bodování

Výstupem projektu bude soubor `xlogin00.zip` obsahující všechny zdrojové soubory a textový soubor `nbody.txt` obsahující textový komentář k projektu. V každém souboru nezapomeňte uvést svůj login! Hodnotit se bude jak funkčnost a správnost implementace, tak textový komentář – ten by měl dostatečně popisovat rozdíly mezi jednotlivými kroky a odpovídat na otázky uvedené v zadání. Při řešení se soustřeďte především na správnost použití CUDA, přesnost výpočtu je závislá na mnoha okolnostech, např. zvoleném výpočtu, pořadí operací apod., a pokud bude v rozumných mezích, nebude hrát velkou roli při hodnocení. Projekt odevzdejte v uvedeném termínu do informačního systému.

[1] `http://en.wikipedia.org/wiki/Nvidia_Tesla`
[2] `http://en.wikipedia.org/wiki/Tesla_(microarchitecture)`