

Paralelní programování na GPU (PCG 2022)

Projekt č. 2: OpenACC

Jiří Jaroš (jarosjir@fit.vutbr.cz)

Termín odevzdání: 16. 12. 2022

1 ÚVOD

V prvním projektu jste si mohli vyzkoušet akceleraci částicového systému pomocí Nvidia CUDA. Cílem tohoto projektu bude tatáž úloha, ale implementace bude probíhat pomocí knihovny OpenACC^{1 2 3} a kompilátoru NVHPC⁴.

Vypracování i opravování bude probíhat na Karolíně.

2 OPENACC NA SUPERPOČÍTAČI KAROLÍNA

Pro připojení na superpočítač Karolína postupujte dle návodu v IS. Pouze při vytváření úlohy zvolte frontu qgpu:

```
[jarosjir@login1.karolina ~]$ qsub -A DD-22-68 -q qgpu -l select=1:ngpus=1,walltime=4:00:00 -I
qsub: waiting for job 1307408.dm2 to start
qsub: job 1307408.dm2 ready
```

```
[jarosjir@cn197.karolina ~]$
```

Následně je nutné načíst moduly v tomto pořadí:

```
m1 NVHPC/22.2 HDF5/1.12.1-NVHPC-22.2
```

¹<https://www.openacc.org/>

²https://www.openacc.org/sites/default/files/inline-files/OpenACC_Programming_Guide_0.pdf

³<https://www.openacc.org/sites/default/files/inline-files/OpenACC.2.7.pdf>

⁴<https://developer.nvidia.com/hpc-sdk>

3 ČÁSTICOVÝ SYSTÉM POMOCÍ OPENACC(10 BODŮ)

Cílem tohoto projektu bude nejprve implementovat a posléze optimalizovat výpočet vzájemného silového působení N těles pomocí knihovny OpenACC. Každé těleso má jistou hmotnost, polohu v prostoru a rychlost. Gravitační síly působící na dané těleso od ostatních těles mají různé směry a jejich výslednice způsobuje změnu rychlosti tohoto tělesa. Pro vektory polohy \mathbf{r} a rychlosti \mathbf{v} platí:

$$\mathbf{r}^{i+1} = \mathbf{r}^i + \mathbf{v}^{i+1} \cdot \Delta t \quad (3.1)$$

$$\mathbf{v}^{i+1} = \mathbf{v}^i + \mathbf{v}_g^{i+1} + \mathbf{v}_c^{i+1} \quad (3.2)$$

kde \mathbf{v}_g^{i+1} je přírůstek rychlosti vzniklý gravitačním působením těles a \mathbf{v}_c^{i+1} je změna rychlosti vlivem kolize s některými tělesy.

Síla působící na těleso je dána vektorovým součtem dílčích sil způsobených gravitačním polem ostatních těles. Dvě tělesa na sebe působí gravitační silou danou:

$$F = \frac{G \cdot m_1 \cdot m_2}{r^2}, \quad (3.3)$$

kde $G = 6.67384 \cdot 10^{-11} \text{Nm}^2\text{kg}^{-2}$ je gravitační konstanta, m_1 a m_2 jsou hmotnosti těles a r je jejich vzdálenost. Rychlost, kterou těleso obdrží díky této síle pak lze vyjádřit jako:

$$\mathbf{v}_g^{i+1} = \frac{\sum \mathbf{F}_j^{i+1}}{m} \cdot \Delta t \quad (3.4)$$

Pokud se tělesa dostanou do příliš blízké vzdálenosti, dané konstantou COLLISION_DISTANCE, dojde k jejich odrazu. Částice si můžete představit jako koule s poloměrem daným polovinou této konstanty. Pro jednoduchost mají všechny tělesa stejný poloměr. Rychlosti dvou těles po odrazu lze určit ze dvou zákonů.

$$v_1 \cdot m_1 + v_2 \cdot m_2 = w_1 \cdot m_1 + w_2 \cdot m_2 \quad (3.5)$$

$$\frac{1}{2} \cdot v_1^2 \cdot m_1 + \frac{1}{2} \cdot v_2^2 \cdot m_2 = \frac{1}{2} \cdot w_1^2 \cdot m_1 + \frac{1}{2} \cdot w_2^2 \cdot m_2 \quad (3.6)$$

kde m_1 a m_2 jsou hmotnosti těles, v_1 a v_2 jsou rychlosti těles před kolizí a w_1 a w_2 jsou rychlosti těles po kolizi. Rovnice 3.5 je zákon o zachování hybnosti a rovnice 3.6 je zákon o zachování kinetické energie. Řešením těchto dvou rovnic o dvou neznámých pro w_1 získáváme novou rychlost tělesa. Jelikož v daném kroku mohou na těleso působit i ostatní tělesa, je potřeba získat pouze rozdíl oproti původní rychlosti, který se na původní rychlost aplikuje později.

Změna rychlosti v daném kroku lze pak vyjádřit jako

$$v_c = w_1 - v_1 \quad (3.7)$$

Pro všechny elementy pak platí

$$\mathbf{v}_c^{i+1} = \sum \mathbf{v}_c^{i+1} \quad (3.8)$$

V každém kroku výpočtu je nutné spočítat změny rychlostí a poloh jednotlivých těles.

3.1 KROK 1: IMPLEMENTACE VZÁJEMNÉHO PŮSOBNÍ TĚLES (2 BODY)

Kostra aplikace je připravena v adresáři `step1` v souborech `main.cpp`, `nbody.cpp`, `nbody.h`. Zde máte vyznačeno, které části je nutné doplnit. V `Makefile` by nemělo být třeba nic měnit (můžete si zvýšit úroveň výpisů pomocí `-Minfo`, nebo přepnout na generování kódu pro CPU pomocí proměnné `ACC`, kde lze také měnit i další parametry akcelerace).

Při implementaci vycházejte ze základní procesorové verze, případně z naivní CUDA verze vytvořené v projektu 1. V tuto chvíli můžete použít pole struktur (AoS) pro jednodušší implementaci.

Zaměřte se na následující body (bude hodnoceno):

- Minimalizaci transferů mezi CPU a GPU mezi iteracemi či voláním kernelů. Doporučuji rozšířit strukturu `Particles` a `Velocities` tak, aby měla metody pro alokaci a dealokaci paměti, a pro transfery na/z GPU (viz cvičení 6).
- Zamyslete se, jak budete ukládat v paměti pozice, hmotnosti a rychlost. Pozor na to, že OpenACC nemá datové typy `float2`, `float3`, `float4`. Jejich tvorba v C++ je však triviální (`struct float4 {float x, y, z, w;}`).
- Navrhněte správně strukturu OpenACC smyček připravených metodách `calculate_gravitation_velocity`, `calculate_collision_velocity` a `update_particle`.
- Pokuste se kód optimalizovat (např. collapse, tile, gang, vector, ...)

Správnost výpočtu je možné ověřit porovnáním výstupního souboru se vzorovým výstupem `sampledata/sampleOutput.h5`, nebo pomocí testů ve složce `tests`. Odchytky v řádech desetin značí, že je ve výpočtu významná chyba. Řádově menší chyby mohou být způsobeny i mírně odlišným výpočtem, dokonce i přeuspořádáním operací. **Průchod testy je nutnou, ne však postačující podmínkou pro udělení bodů z každého úkolu.**

Po ověření správnosti vyplňte tabulku v souboru `nbody.txt` a odpovězte na dotazy. Tabulka bude obsahovat naměřenou dobu běhů simulace pro různé velikosti dat.

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 přidáné metriky a na jejich základě optimalizujte svůj kód.

3.2 KROK 2: IMPLEMENTACE VZÁJEMNÉHO PŮSOBNÍ TĚLES (2 BODY)

Zkopírujte celý adresář `step1` do nového adresáře `step2`. Vytvořte novou funkci `calculate_velocity` s vhodným rozhraním, který bude implementovat funkčnost všech předchozích kernelů. Zde na GPU alokujte vše dvakrát, v každém kroku výpočtu pak použijte jednu kopii dat jako vstupy (`p_in`) a druhou jako výstupy (`p_out`). V každém dalším kroku pak tyto dvě kopie vždy prohod'te. Tím bude možné některé výpočty zjednodušit a dále optimalizovat.

Pomocí profilování zjistíte rozdíly mezi implementacemi v kroku 1 a 2 a tyto rozdíly popište v souboru nbody.txt. Popřemýšlejte jestli je výhodnější použít Pole struktur (AoS) nebo strukturu polí (AoS)

Cílem tohoto kroku je dostat co nejvyšší výkon, při udržení rozumně elegantního kódu (OpenACC je primárně o snadném návrhu, čitelnosti, udržitelnosti...). Tedy, pokud potřebuji 5 pomocných proměnných, abyste různě otáčeli data v rámci jednoho kernelu, tak to asi není to pravé ořechové. Projekty opravuji ručně, proto mi komentáře mohou pomoci k správnému pochopení kódu.

3.3 KROK 3: VÝPOČET TĚŽIŠTĚ (2 BODY)

Opět zkopírujte celý adresář step2 do nového adresáře step3. V tomto kroku je vaším úkolem doplnit kód pro výpočet těžiště částicového systému na GPU. Jako inspirace Vám může sloužit CPU varianta. V CUDA variantě jsme sázeli na redukci ve sdílené paměti a použití zámků. To v OpenACC nebude tak jednoduché, protože OpenACC je navrženo jako lockfree jazyk. Jaké tedy máte možnosti, pokud nechceme dělat výpočet sekvenčně:

- bariera - bariery lze udělat pouze mezi kernely - tedy není možné provést uvnitř acc parallel.
- zámký - nejsou v jazyce podporovány, ale můžete si je implementovat pomocí atomických operací s dovětkem capture.
- kritická sekce - neexistuje, tedy update více položek atomicky nelze provést. Jedině vlastní zámeček.
- redukce - funguje pouze nad základními datovými typy, ne nad float4.

Inspiraci můžete nalézt v paralelní redukci v CUDA, jen si představte, že nemáte sdílenou paměť ale pouze tu globální. Pokud to bude nutné, máte povoleno alokovat další paměť na GPU. Dělejte to ale pouze 1x v celém kódu, ne při každé iteraci.

Pokud se vám povede navrhnout více verzí, a nevíte která je nejlepší, klidně je nechte v řešení, ale dejte jim nějaké jiné jméno např. centerOfMassGPU_ver1. Při hodnocení správnosti se budu dívat na tu první, ale mohu vám dát nějaký další bod za zajímavý návrh.

Pozor: Výpočet těžiště musí fungovat pro libovolný počet částic, nejenom mocninu dvou.

3.4 KROK 4: PŘEKRYTÍ VÝPOČTU POZIC, TĚŽIŠTĚ A UKLÁDÁNÍ DAT NA DISK (2 BODY)

Opět zkopírujte celý adresář step3 do nového adresáře step4. Nejdříve doplňte kód pro zápis do souboru v každé writeIntesity iteraci. Funkce pro zápis jsou writeParticleData a writeComData z modulu h5Helper. Funkce writeComData vyžaduje explicitně předat hodnoty těžiště (spočteno na GPU) a číslo záznamu. Funkce writeParticleData si vystačí s číslem záznamu (interně využívá MemDesc). Několik pravidel pro zápis:

- zapisujte v každé n -té iteraci, kde n je rovno writeIntesity, a vždy v iteraci 0.

- V iteraci n počítejte těžiště z hodnot částic kroku n a zapisujte hodnoty částic z kroku n . Současně počítejte nové hodnoty kroku $n + 1$ (v iteraci 0 počítejte těžiště a zapisujte vstupní hodnoty).
- `writeIntesity` rovno 0 znamená že se vůbec nezapisuje (zápis jen po skončení simulace) a vyžaduje speciální ošetření.

Pokud se vám nepovede udělat vše, implementuje alespoň část (např. výpočet (polohy + těžiště) - transfer). Náповědu hledejte v OpenACC 2.7 API, kapitola 2.16⁵

3.5 KROK 5: ANALÝZA VÝKONU (2 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.h5
```

Naměřené časy porovnejte se paralelní implementací CPU (stačí přeložit s `-acc=multicore`) verze a spočtete zrychlení. Pokud máte naměřeny hodnoty i pro CUDA verzi, připište i tyto hodnoty. Od jakého počtu částic se vyplatí použít grafickou kartu?

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 a jméno (přepište ten můj)! 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 soustřed'te především na správnost použití OpenACC, 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. Projekty hodnotím ručně :) Projekt odevzdejte v uvedeném termínu do informačního systému.

⁵<https://www.openacc.org/sites/default/files/inline-files/OpenACC.2.7.pdf>