Семинар 5: Распараллеливание по взаимодействующим парам частиц

Для того, чтобы избежать двойного расчёта сил для парных потенциалов на ГП, можно использовать другой алгоритм, в котором каждый поток рассчитывает одно значение потенциала для пары частиц. При этом силы, разнонаправленно действующие на взаимодействующие частицы, вычисляются лишь один раз, а полученные значения сохраняются в различных ячейках глобальной памяти ГП. Затем все силы, действующие на отдельную частицу, суммируются для вычисления результирующей силы. Использование этого подхода требует дополнительных обращений к памяти и разработки отдельной процедуры, выполняющей итоговое суммирование. Тем не менее, он позволяет ускорить выполнение моделирования, если количество частиц N того же порядка, что и число АЛУ и / или если вычисление парных потенциалов ресурсо-затратно. В следующем фрагменте псевдокода алгоритма для расчёта сил, число потоков P равно количеству взаимодействующих пар (только для одного слагаемого потенциальной энергии), и каждый поток вычисляет силы для отдельной пары частиц:

Алгоритм 2: Расчёт сил взаимодействия между частицами (реализация для ГП с использованием подхода распараллеливания по взаимодействующим парам частиц).

1: $p \leftarrow$ индекс потока вычислений на ГП {равен индексу пары}

2: $pair \Leftarrow pairs[p]$

3: $par \Leftarrow PairsParameters[p]$ {параметры одной пары частиц p}

4: $i \leftarrow pair.i$ {i-тая частица пары}

5: $j \leftarrow pair.j$ {j-тая частица пары}

6: $shift_i \leftarrow pair.shift_i$ {позиция в массиве сил для i-той частицы}

7: $shift_j \leftarrow pair.shift_j$ {позиция в массиве сил для j-той частицы}

8: $\vec{r}_i \leftleftarrows \vec{r}[i]$ {координаты i-той частицы}

9: $\vec{r}_j \leftleftarrows \vec{r}[j]$ {координаты j-той частицы}

10: $\vec{r} \leftarrow \vec{r}_i - \vec{r}_j$

11: $\vec{df} \leftarrow \texttt{force}(\vec{r}, par)$

12: $\vec{df} \Rightarrow \vec{F}[i][shift_i]$ {сохранение значение силы для i-той частицы}

13: $-\vec{df} \Rightarrow \vec{F}[j][shift_j]$ {сохранение значение силы для j-той частицы}

Каждому потоку соответствует одна пара с тем же индексом p, который идентифицирует поток. Потоком считывается информация о потенциале из вектора $pairs$ и о значениях постоянных параметров par из вектора $PairsParameters$, определяются взаимодействующие частицы i и j их координаты (r_i и r_j), а также адреса глобальной памяти, по которым необходимо сохранять значения сил ($shift_i$ и $shift_j$). В массиве F значения сил сохраняются на позициях, соответствующих индексу частицы i и j (i-тая или j-тая строка) и параметрам $shift_i$ и $shift_j$ (столбцы). В массиве $pairs$ итоговые позиции i и $shift_i$ для i-той частицы должны быть индивидуальны для каждой из частиц с тем, чтобы вычисленные значения сил сохранялись в глобальной памяти ГП по различным адресам. Это позволяет избежать конфликтов памяти, но требует разработки дополнительного ядра, выполняющего суммирование всех сил, действующих на определённую частицу, которые хранятся в массиве F (Алгоритм 4):

Алгоритм 3: Ядро сложения для вычисления результирующей силы.

1: $i \leftarrow$ индекс потока вычислений на ГП {равен индексу частицы}

2: $\vec{f_i} \leftarrow 0$ {результирующая сила (определяется одним слагаемым потенциала)}

3: $P_i \leftleftarrows P_p[i]$ {число пар с участием i-той частицы}

4: **for** $p = 0$ to $P_i - 1$ **do**

5: $\quad \vec{df} \leftleftarrows \vec{F}[i][p]$

6: $\quad \vec{f_i} \leftarrow \vec{f_i} + \vec{df}$

7: **end for**

8: Output: $\vec{f_i}$

Переменная P_i используется для счёта частиц, парных по отношению к i-той (для одной из потенциальных функций). Результирующая сила для i-той частицы (f_i) рассчитывается суммированием всех сил, вычисленных до этого ($F[i][p]$, строка 12-13 в алгоритме 4). Эта часть программы может быть встроена в интегрирующее ядро для сокращения количества вычислительных ядер. На ГП новой архитектуры Fermi (от NVIDIA) наличие атомарного добавления `atomicAdd(...)` для вещественных величин позволяет производить сложение вычисленных значений сил при выполнении ядра, вычисляющего силы [**?**]. Это устраняет барьер производительности, связанный с многочисленными обращениями к памяти.

Списки Верле: При использовании подхода распараллеливания по парам взаимодействующих частиц создание списка Верле сводится к формированию вектора $pairs$ из всех взаимодействующих пар частиц в рамках одного слагаемого потенциала. Формирование вектора на ГП усложняется тем, что точная позиция в списке, куда добавляется информация о новой взаимодействующей паре, изначально неизвестна. Одно из возможных решений этой проблемы - использование функции `atomicAdd(...)` для целочисленных величин из библиотеки CUDA Software Development Kit [**?**], которая позволяет складывать целые числа, хранящиеся в глобальной памяти ГП, избегая конфликтов обращения к памяти, даже в случае, если множество потоков одновременно работает с информацией, хранящейся по одному и тому же адресу памяти. Параллельная работа потоков, выделяющих взаимодействующие пары и последовательно записывающих их, может привести к формированию списка Верле, не отсортированного в соответствии с порядковыми номерами частиц. Это может привести к неэффективному использованию кэш-памяти при чтении координат и несоблюдению условий коалесинга. Упорядочить лист Верле можно, если после его составления применить сортировку по индексу частицы (на ГП или ЦП). Другой вариант заключается в вычислении расстояний между частицами на ГП, копировании их в DRAM память ЦП и последующем составлении нового списка Верле. В силу того, что при использовании динамики Ланжевена вода представлена неявно, обновлять список Верле

можно редко. Из-за этого, использование даже не оптимизированного кода не приведёт к серьёзным задержкам в расчёте.