Московский государственный университет имени
М.В.Ломоносова
Факультет вычислительной математики и кибернетики

Суперкомпьютерное моделирование и технологии

Решение трёхмерного гиперболического уравнения в прямоугольном параллелепипеде с использованием графических процессоров

DVMH

Зыкин Ярослав Ильич
628 группа

17 декабря 2022 г.

Содержание

1 Постановка задачи

Для трёхмерного гиперболического уравнения в прямоугольном параллелепипеде необходимо расширить программу, полученную в результате выполнения третьего практического задания, с помощью системы DVMH[1] с целью ускорения её выполнения на графических процессорах.

Необходимо исследовать масштабируемость разработанной программы по числу процессов, нитей и графических процессоров.

Также необходимо исследовать время работы отдельных частей программы на графических процессорах.

[1] http://dvm-system.org

2 Программная реализация

При подготовке исходной программы третьего задания к использованию системы DVMH были сделаны следующие действия:

- Программа была переписана с языка C++ на язык C, так как DVMH не поддерживает C++.
- Плоский массив был заменён на трёхмерный для упрощения распределения массивов средствами DVMH.

Особенности решения задачи с использованием технолии DVMH

- Для реализации блочного разбиения используюься средства DVMH распределения трёхмерных массивов с использованием теневых граней размера 1 по каждому из измерений.
- Подсчёт каждого временного слоя заключается в отдельный dvm-регион.
- Каждый цикл (кроме цикла по числу шагов) распараллеливается директивой dvm parallel.
- Цикл подсчёта нового временного слоя использует директиву shadow_renew для обновления теневых граней.
- Цикл подсчёта ошибки на слое использует директиву reduction(max()) для получения максимального значения ошибки при параллельном выполнении.
- Перед выводом ошибки подсчёта функции на временном слое используется директива get_actual, которая позволяет получить текущее значение ошибки (если это значение вычислилось на графическом процессоре, то после этой директивы оно окажется в памяти центрального процессора).

3 Эксперименты

Для исследования масштабируемости разработанной программы были проведены расчёты для пространственной сетки размера 768^3 на 1, 20 и 40 процессов с разными конфигурациями по количеству нитей центрального процессов и графических процессов. Для сравнения результатов был проведён запуск последовательной программы, написанной в рамках третьего практического занятия.

Все полученные результаты приведены в таблице 3 и на графиках ускорения 1 и 2

В качестве лучшей версии была взята конфигурация с одним графическим процессором и восемью нитями центрального процессора в соотношении 0.9 у GPU к 0.1 у всех нитей. Полученные результаты приведены в последнем столбце таблицы 3.

По полученным данным можно сделать следующие выводы.

- Разработанная программа работает быстрее, чем программа, разработанная в рамках третьего задания. Особенно это заметно при сравнении с запусками с использованием графических процессоров.
- Запуск программы с помощью системы DVMH приводит к очень большому разбросу по времени для разных запуков (достигает десятка раз).
- Достаточного для усреднения количества данных по работе программы на 40 процессах получить не удалось из-за большой нагрузки на суперкомпьютер.
- Увеличение количества нитей не влияет на скорость работы программы.
- При использовании 20 процессов время работы сильно уменьшается, по сравнению со временем работы при использовании 1 процесса.
- При использовании 40 процессов скорость работы программы уменьшается. Это может быть связано с тем, что работа производится на разных хостах (с помощью системы DVMH можно гарантировать, что при запусках на 20 процессов используется только один хост) и, соответственно, накладные расходы на передачу данных существенно выше, чем внутри одного хоста.
- При использовании системы DVMH большую часть времени счёта программы занимает работа самой системы, поэтому результаты для 2 и 4 графических процессоров при запуске на 20 и 40 процессах получить не удалось (был превышен лимит в 30 минут, а запустить с большим времем (в очередь normal) через систему DVMH возможности найдено не было).

Количество процессов	Было	Количество нитей				Количество GPU			8T+1G
		1	2	4	8	1	2	4	
1	550.827	494.815	498.064	494.376	490.379	7.340	2.441	1.871	4.676
20		9.899	9.124	7.423	7.476	2.000	–	–	2.987
40		15.358	54.130	34.125	41.634	1.185	–	–	–

Таблица 1: Время выполнения программы

Анализ работы графических процессоров

Для анализа загрузки графических процессоров была выбрана конфигурация с единственным графическим процессом, потому что только эта конфигурация позволила получить все данные. При этом рассмотрены будет вариант запуска на 20 процессах, потому что в этом случае, в отличие от 1 процесса, есть накладные расходы, связанные с пересылкой данных, и данные о работе программы проще анализировать, чем в случае 40 процессов. Рассматриваемыми характеристиками будут:

1. Время, затраченное на обмен сообщениями на центральных процессорах.

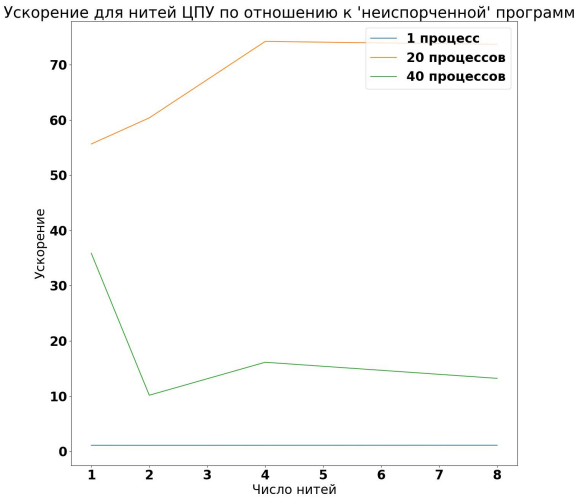

Рис. 1:

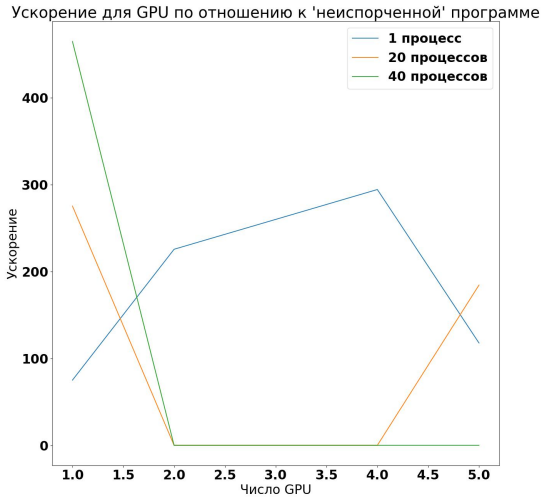

Рис. 2:

2. Время на копирование теневых граней с графического процессора на центральный.
3. Время на копирование теневых граней с центрального процессора на графический.
4. Время на копирование данных при входе в вычислительный регион.
5. Время работы параллельного цикла.
6. Время, затраченное на операцию редукции.
7. Время, затраченное на блокировку страниц памяти основного процесса.
8. Время полезной нагрузки графического процессора (п.4 и п.5)
9. Время накладных расходов графического процессора (п.2, п.3, п.6, п.7).

Усреднённые результаты можно посмотреть в таблице 3, полные результаты приведены на графиках 3 и 4

	обмен	теневые0	теневые1	регион	цикл	редукция	блок	полезная	расходы
min	0.1011	0.0058	0.0229	0.4206	0.4259	0.0383	0.7278	0.8465	0.8894
avg	0.3058	0.0133	0.1231	0.4586	0.5836	0.1001	0.7438	1.0422	0.9802
max	0.5893	0.0277	0.2512	0.4965	0.6993	0.1407	0.7639	1.1610	1.0721

Таблица 2: Профилирование программы

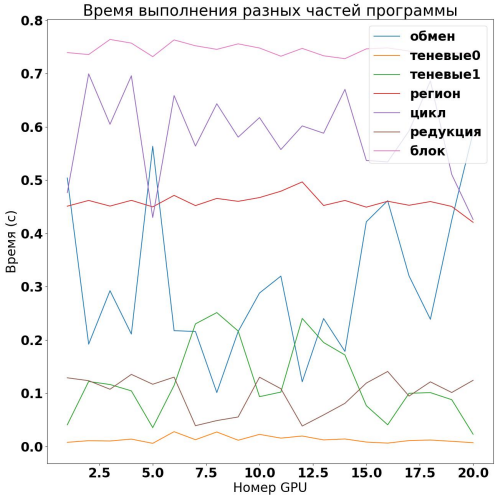
Время выполнения разных частей программы
обмен
теневые0
теневые1
регион
цикл
редукция
блок
Время (с)
Номер GPU

Рис. 3:

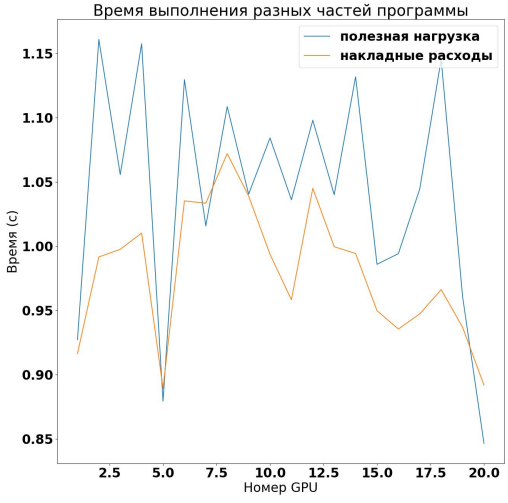
Время выполнения разных частей программы
полезная нагрузка
накладные расходы
Время (с)
Номер GPU

Рис. 4: