Параллельные и распределенные вычисления. Задание 4 “ Игра *Жизнь* ”.

Старченко Владимир

Основная часть

Поставленная задача

Реализуйте параллельную версию игры "Жизнь"с использованием MPI. Обоснуйте выбранную стратегию распараллеливания. Выполните тестирование параллельной программы с помощью предоставленного сервиса.

Обоснование решения

В данной задаче нужно было раздать разные участки обсчитываемого поля. Есть разные разбиения. Я выбирал между квадратами и прямоугольниками. Теоретически, квадратами выгоднее, так как оно имеет более хорошее отношение площади разбитого кусочка к периметру кусочка. Однако оно имеет некоторые недостатки. Не всегда можно удачно разбить (неподходящее количество потоков). В конкретно этом случае нам гарантируется, что всё будет хорошо, однако на практике так будет не всегда и нужно смотреть по конкретной задаче. Другое разбиение - прямоугольниками. Оно проще реализуется и из-за этого было выбрано мной в качестве начального варианта.

В данном подходе мы разбиваем плоскость на прямоугольники. Каждый прямоугольник граничит с соседним. Так как для вычисления значений в ячейке требуется знать значения восьми соседних, все внутренние ячейки могут обсчитываться независимо от других прямоугольников. Чтобы посчитать ячейки на граничных полосках, нужно обменяться сообщениями с процессами, отвечающими за соседние прямоугольники. Таким образом, на каждой итерации процесс должен отсылать и принимать значения границ соседних прямоугольников.

Описание реализации

В начале я реализовал ровно ту последовательность действий, что описана выше. Основной поток считывает файл. Рассылает scatter-ом полоски другим потокам. В начале каждой итерации обмениваемся значениями границ. Ждем приема сообщений. обсчитываем прямоугольник, переходим к следующей итерации. В конце собираем все значения в основном потоке gather-ом.

Результаты

https://everest.distcomp.org/jobs/5a1ab6d7310000670e8b30e5

Анализ решения

Это работает, но недостаточно быстро (0.7). Можно заметить, что (как было сказано выше) все внутренние ячейки могут обсчитываться независимо от других прямоугольников. При этом время, которое уходит на обмен сообщений довольно большое и процессы простаивают. Следовательно, можно оптимизировать работу. Вызвать неблокирующие функции обмена сообщениями и постовить считаться все внутренние ячейки. После окончания дождаться прихода сообщений и посчитать оставшиеся граничные ячейки.

Такая последовательность действий дает значительное ускорение (1.0), потому что при достаточно большом количестве внутренних ячеек время их обсчета будет больше времяни обмена сообщениями. Следовательно, на обмен сообщениями не будет тратиться почти ничего. Следовательно, эффективность может быть примерно равна 1.

Данное решение не слишком оптимальное в смысле реализации однопоточной версии. Поэтому соответствующая ему многопоточная реализация также неоптимальна. Можно было бы ещё различнами способами разгонять многопоточную реализацию, однако в этом случае пропала бы наглядность при сравнении эффективности и ускорения.

Контрольные вопросы

Вопрос 1.

Как зависят ускорение и эффективность от числа процессов? Почему? (вес 0.3)

Мы видим две сильно различающиеся картины. При маленьком размере поля и при большом.

При маленьком:

Test 1 (N=576, iter=10000):

Процессы	Время	Ускорение	Эффективность
4	33.20	4.10	1.02
9	15.00	9.07	1.01
16	8.82	15.42	0.96
36	4.51	30.16	0.84
64	3.09	44.01	0.69

Эффективность меньше 1 и падает при увеличении количества процесов. Это говорит о том, что при маленьком количестве данных внутренние ячейки обсчитываются быстрее, чем приходят сообщения. Процессам приходится простаивать, прежде чем пересчитать граничные ячейки. Соответственно ускорение растет медленнее, чем линейно.

При большом:

Test 3 (N=3456, iter=10000):

Процессы	Время	Ускорение	Эффективность
64	72.23	66.86	1.04

Так как количество непараллельного кода мало и процессы просто так не простаивают, Эффективность примерно равна единице. Соответственно ускорение линейное. То, что эффективность чуть больше едници объясняется особенностью реализации и неточностью при замерах (Цифры немного оличаются от запуска к запуску).

С теоретической точки зрения, если мы зафиксируем количество итераций, зафиксируем размер поля и будем увеличивать количество процессов эффективность будет падать по выше изложенным причинам.

Вопрос 2.

Как зависит ускорение от числа итераций (при фиксированном размере поля и числе процессов)? Почему? (вес 0.3)

При сравнении показанных ниже таблиц можно увидеть, что при увеличении количества итераций ускорение увеличивается. Это объясняется тем, что доля непараллельного кода уменьшается. Действительно, при малом количестве итераций время на рассылку изначальных данных (scatter) и сбор из обратно (gather) занимают большое время. И при большом количестве итераций им можно совсем пренебречь.

Test 1 (N=576, iter=10000):

Процессы	Время	Ускорение	Эффективность
4	33.20	4.10	1.02
9	15.00	9.07	1.01
16	8.82	15.42	0.96
36	4.51	30.16	0.84
64	3.09	44.01	0.69

Test 2 (N=576, iter=100000):

Процессы	Время	Ускорение	Эффективность
16	80.63	16.57	1.04
36	37.70	35.44	0.98
64	22.16	60.29	0.94

Вопрос 3.

Как зависит ускорение от размера поля (при фиксированном числе итераций и числе процессов)? Почему? (вес 0.4)

Частично ответ на этот вопрос был дан в пункте 1. Чем меньше размер поля, тем меньше эффективность из-за простаивания процессов. Вплоть до того что количество прямоугольников может быть равно количеству процессов и почти всё время процессы будут ждать. Еси бы будем увеличивать размер поля, эффективность будет возрастать до значения единицы. При этом количество времяни простоя станет равным нулю и дальнейшее увеличение размера поля не будет влиять на эффективность, так как она не поднимется больше 1. (С точностью до отклонений вызванных особенностями реализации и измерением времени.) Соответственно ускорение зависит линейно от эффективности.

Test 1 (N=576, iter=10000):

Процессы	Время	Ускорение	Эффективность
4	33.20	4.10	1.02
9	15.00	9.07	1.01
16	8.82	15.42	0.96
36	4.51	30.16	0.84
64	3.09	44.01	0.69

Test 3 (N=3456, iter=10000):

Процессы	Время	Ускорение	Эффективность
64	72.23	66.86	1.04