# Постановка задачи

Исследовать эффективность (время, ускорение) на кластере мэи параллельных алгоритмов сортировки больших числовых массивов

Дано матрица M(nxn).

1. Разработать MPI-алгоритм решения задачи, в которых можно варьировать:

-n-размер матрицы

-степень распараллеливания, как на уровне строк, так и на уровне мелкозернистого распараллеливания в строке.

2) Исследовать эффективность алгоритмов, варьируя размер матрицы

n=1000-5000 или

10 000-20 000 – 50 000

Матрицы заполняются случайным образом, используя генератор случайных чисел.

Предполагается сравнить следующие алгоритмы сортировки:

1. Быстрая сортировка
2. Сортировка слиянием

ИТЕРАЦИЯ 2(упрощение)

1. Узлов на кластере n(n<=16)
2. Размер сортируемой строки 5000
3. Количество сортируемых строк-k, изменяется от k=n(вообще зафиксировано)
4. Алгоритмы сортировки на узле: быстрой, другой+(?)2 варианта другой
5. Предоставить графики времени от количества узлов, ускорения

Формат отчета – обычный, по сути лаба

# Вопросы где он опять может устроить блиц-опрос:

***Степенью параллелизма*** этапа численного алгоритма называется число операций, которые на данном этапе можно выполнять параллельно (ширина графа, построенного описанным выше способом).

От чис­ла про­цес­со­ров за­ви­сит вре­мя, не­об­хо­ди­мое для за­вер­ше­ния вы­чис­ле­ний. На­при­мер, ес­ли n=1000, и чис­ло про­цес­со­ров *p* так­же рав­но 1000, то все сум­мы тео­ре­ти­че­ски мож­но вы­чис­лить за один вре­мен­ной шаг, од­на­ко при p=10 по­тре­бу­ет­ся 100временны́х ша­гов.

**T=n/p это время выполнения(n-степень, p-число процессоров)**

|  |  |
| --- | --- |
| S_p=\frac{T_1}{T_p}, | (4) |

***Ускорением параллельного алгоритма называется отношение:***

*S\_p=\frac{T\_1}{T\_p},*

*(4)*

*где T\_p — время вычисления задачи на p процессорах.Заметим, что в определении подразумеваются действительные времена вычислений. Это делает определение более полезным на прак­ти­ке, но затрудняет его использование в том случае, если требуемые времена неизвестны. Для идеальной вычислительной системы и идеальной параллельной реализации алгоритма его ускорение равно средней степени параллелизма.*

С ускорением связана эффективность параллельного алгоритма. ***Эффективностью параллельного алгоритма(p-число процессоров, Sp-ускорение)*** называется величина:

|  |  |
| --- | --- |
| E_p=\frac{S_p}{p}. | (5) |

По определению, E_1=1. Теоретически должно быть  S_p\leqslant p и E_p\leqslant 1. Если алгоритм достигает максимального ускорения (S_p=p), то E_p=1. На практике эффективность убывает при увеличении числа процессоров.

Иногда бывают случаи, когда E_p>1  («суперлинейное ускорение»). Эта аномалия вызвана, чаще всего, двумя причинами:

* В качестве последовательного алгоритма был применён не самый быстрый алгоритм из доступных.
* С увеличением количества вычислителей растёт суммарный объ­ём их оперативной и кэш памяти. Поэтому всё большая часть дан­ных задачи умещается в оперативной памяти и не требует подкачки с диска, или (чаще всего) умещается в кэше («аггрегация кэшей»). В таких случаях для точного измерения эффективности реализо­ванного алгоритма следует уменьшать объём кэш памяти каждого вычислителя обратно пропорционально числу вычислителей.

***Средней степенью параллелизма****численного алгоритма называется отношение общего числа операций алгоритма к числу его этапов*. Очевидно, для алгоритма сдваивания средняя степень параллелизма равна:

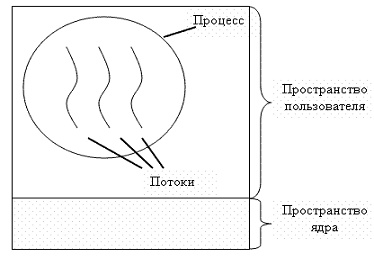
|  |  |
| --- | --- |
| S=\frac{n-1}{\log_2n}. | (3) |

Со степенью параллелизма также связано понятие *зернистости*. ***Крупнозер­нистость*** задачи означает наличие в ней больших независимых подзадач, которые можно обрабатывать параллельно. Примером может служить зада­ча решения нескольких различных больших систем линейных уравнений, ре­шения которых комбинируются на более поздних стадиях вычислительного процесса. ***Мелкозернистость*** соответствует возможности параллельного выполнения малых подзадач. Так, для сложения двух векторов подзадачей является сложение компонент, имеющих одинаковый номер. Крупнозерни­стые алгоритмы сложно распараллелить на большом числе процессоров.

Из курса теории операционных систем известно, что процесс является *динамическим объектом*, описывающим выполнение программы. Процессу выделяются системные ресурсы: закрытое адресное *пространство*, семафоры, коммуникационные порты, файлы и т.д. Процесс характеризуется текущим состоянием (выполнение, ожидание, готовность и т.д.).

#### Потоки

Классический процесс содержит в своем адресном пространстве одну программу. Однако во многих ситуациях целесообразно поддерживать в *едином адресном пространстве* процесса несколько выполняющихся программ (потоков команд или просто *потоков* ), работающих с общими данными и ресурсами.



http://iproc.ru/parallel-programming/lection-3/