MPI: перемножение матрицы на вектор

Николай Игоревич Хохлов

МФТИ, Долгопрудный

12 апреля 2017 г.

Постановка задачи

- Пусть имеются матрица A размером $m \times n$.
- Требуется вычислить $c = A \times b$.
- Где b – вектор размера n, c – вектор размера m.
- Имеется p процессов.

Последовательный алгоритм

$$\begin{pmatrix} a_{0,0} & \cdots & a_{0,j} & \cdots & a_{0,n-1} \\ & & \cdots & & \\ \mathbf{a_{i,0}} & \cdots & \mathbf{a_{i,j}} & \cdots & \mathbf{a_{i,n-1}} \\ & & \cdots & & \\ a_{m-1,0} & \cdots & a_{m-1,j} & \cdots & a_{m-1,n-1} \end{pmatrix} \times \begin{pmatrix} b_{0,0} \\ . \\ b_{i,0} \\ . \\ b_{n-1,0} \end{pmatrix} = \begin{pmatrix} c_{0,0} \\ . \\ c_{i,0} \\ . \\ c_{m-1,0} \end{pmatrix}.$$

Последовательный алгоритм

Algorithm 1 Последовательный алгоритм

for $i = 0, ..., m-1$ **do**
  $c_i = 0$
  for $j = 0, ..., n-1$ **do**
    $c_i = c_i + A_{i,j} * b_j$
  end for
end for

Последовательный алгоритм

- Два вложенных цикла. Каждое значение c_i вычисляется как скалярное произведение между i строкой матрицы A и вектором b.
- Независимые вычисления. Каждый элемент c_i может вычисляться независимо.
- Независимость по данным. Количество операций одинаковое для любых наборов данных (исключения для разреженных матриц).
- Регулярная структура организации данных.
- Параллельный алгоритм строится по модели SPMD (Simgle Programm – Multiple Data).
- Сложность $O(mn)$.

Параллельный алгоритм: декомпозиция матрицы

- Параллелизм строится на декомпозиции матрицы A.
- Три подхода к декомпозиции:
  - Построчная декомпозиция (rowwise).
    - Один процесс отвечает за набор последовательно идущих строк матрицы A.
    - В простейшем случае m/p строк.
  - Поколоночная декомпозиция (columnwise).
    - Один процесс отвечает за набор последовательно идущих колонок матрицы A.
    - В простейшем случае n/p колонок.
  - Блочная декомпозиция (checkerboard).
    - Один процесс отвечает за блок матрицы A.

Параллельный алгоритм: декомпозиция векторов

- Два подхода к декомпозиции векторов:
  - Дубликаты на всех процессах.
  - Блочная декомпозиция.
- Приемлемо ли хранить все вектора?:
  - Хранение A занимает mn.
  - Хранение b занимает n.
  - Хранение c занимает m.

Параллельный алгоритм: стратегии параллелизации

- Построчная декомпозиция мматрицы, дубликаты векторов.
- Поколоночная декомпозиция матрицы, блочная декопмозиция векторов.
- Блочная декомпозиция матрицы, блочная декопмозиция векторов.

Параллельный алгоритм: построчная декомпозиция

- Каждая строка матрицы A выделяется в отдельную задачу.
- Вектора b и c реплицируются на каждую задачу.
- Задача i имеет строку i и копию b:
  - вычисление c_i как скалярное произведение A_i и b.
- Для репликации всего вектора c требуются обмены:
  - коллективная операция типа all-gather.
- Каждый процесс работает с набором последовательно идущих строк.

Параллельный алгоритм: построчная декомпозиция

Анализ сложности.

- Пусть для простоты $m = n$, сложность последовательного алгоритма $O(n^2)$.
- Если число процессов p, то каждый процесс имеет как минимум $\lceil n/p \rceil$ строк матрицы.
  - Сложность без пересылок на процесс $O(n^2/p)$.
  - Операция типа all-gather требует от каждого процесса пересылок $\lceil log_2 p \rceil$:
    - Число переселаемых элементов $n(p-1)/p$.
    - Сложность пересылки $O(n + log_2 p)$.
  - Общая сложность $O(n^2/p + n + log_2 p)$.

Параллельный алгоритм: поколоночная декомпозиция

- Каждая колонка матрицы A выделяется в отдельную задачу.
- Вектора b реплицируются на каждую задачу.
- Задача i перемножает i колонку на b_i:
  - Получается вектор полного размера, но с частичным результатом.
- Для подстчета c требуются обмены:
  - Каждый частичный результат j для задачи i передается задаче j.
  - Обмены типа all-to-all.
- Каждый процесс суммирует значения.

Параллельный алгоритм: поколоночная декомпозиция

Анализ сложности.

- Пусть для простоты $m = n$, сложность последовательного алгоритма $O(n^2)$.
- Если число процессов p, то каждый процесс имеет как минимум $\lceil n/p \rceil$ строк матрицы и такое же количество элементов b и c.
- Сложность без пересылок на процесс $O(n^2/p)$.
- Сложность суммирования $O(n)$.
- Операция типа all-to-all:
  - Вариант 1: требует $\lceil log_2 p \rceil$ пересылок:
    - На каждом шаге процесс отправляет n/p значений и принимает столько же.
    - Сложность $O(nlog_2 p)$.
  - Вариант 2: каждый процесс отсылает сообщения всем соседям:
    - Сложность $O(n + p)$.
- Общая сложность $O(n^2/p + nlog_2 p)$ или $O(n^2/p + n + p)$.

Параллельный алгоритм: блочная декомпозиция

- Каждый элемент матрицы A выделяется в отдельную задачу.
- Задача i, j перемножает $a_{i,j}$ на b_j, получает $d_{i,j}$.
- Элемент c_i получается как $\sum_j d_{i,j}$.
  - Требуются обмены.
- Каждый процесс отвечает за набор задач в виде последовательного блока матрицы.

Параллельный алгоритм: блочная декомпозиция

Последовательность алгоритма

- Двумерная декомпозиция, каждый процесс получает блок $A_{i,j}$.
- Пусть вектор b распределен между первой колонкой двумерной декомпозиции.
- Шаг 1: рассылка вектора b так, чтобы каждый процесс имел соответствующий блок b_j.
- Шаг 2: переменожение $A_{i,j}$ на b_j.
- Шаг 3: каждая колонка двумерной декомпозиции делает sum-reduction.

Параллельный алгоритм: блочная декомпозиция

Анализ сложности

- Пусть двумерная декомпозиции размером $k \times l$.
- Вектор b распределен между k процессами у первой колонки сетки декомпозиции.
- Рассылка вектора b между l колонками декомпозиции $O(nlog_2 p/\sqrt{p})$.
- Для каждой колонки и строчки декомпозиции создается свой коммуникатор.
- Для простоты $n = m$ и p квадрат целого числа.
- Каждый процесс отвечает за блок размера $\lceil n/\sqrt{p} \rceil \times \lceil n/\sqrt{p} \rceil$, сложность переменожения $O(n^2/p)$.
- Финальная редукция суммирования $O(nlog_2 p/\sqrt{p})$.

Метод сопряженных градиентов, построить сравнительные графики ускорения

- Построчная – 0.5.
- Поколоночная – 0.5.
- Блочная – 1.