Описание и исследование реализации алгоритма градиентного спуска со случайным начальным приближением для задачи распределения вычислительной нагрузки в многопроцессорной вычислительной системе.

Михайлов Д. М. 213 группа

Общее описание алгоритма

В целом реализация алгоритма полностью удовлетворяет условию. Целочисленная постоянная const int number_of_sets задает количество обрабатываемых наборов данных. Целочисленная постоянная const int tries_count задает количество попыток анализа одного набора. Для каждой попытки заново выбирается начальное приближение, которое затем обрабатывается алгоритмом. Как результат выбирается вектор распределения задач по процессорам с наихудшей суммой интенсивностей обмена. Тем самым выбирается «безопасный» вариант (как и описано в условии). Результат обработки набора данных записываются в вектор results[worst_result_sum], который затем выводится в файл. Функция void init_approx(vector<int> &, vector<int> &, vector<int> &) отвечает за составление начального приближения вектора распределения задач по процессорам (vector<int> tasks_on_cpu). Функция vector<int> approx_handler(vector<int> &, vector<int> &, vector<int> &, vector<vector<int>> &) представляет собой непосредственно реализацию алгоритма. В этой функции выполняются операции a, b, c и выбирается наилучшее решение на данном этапе работы. После того, как не удалось найти ни одно «хорошее» решение (суммы интенсивностей обмена до и после выполнений операций a, b, c идентичны), поиск решения заканчивается. Функции vector<int> operation_k(vector<int> &, vector<int> &, vector<int> &, vector<vector<int>> &), где k = {a, b, c} являются реализациями операций a, b, и c из условия. Результат работы для каждого набора данных выводится с помощью функции void write_result_into_file(int, unsigned int, unsigned int, ofstream &, vector<vector<int>> &, int) в файл “alg_output.txt”. В указанном файле можно видеть для каждого набора:

- Номер набора
- Интенсивность обмена в худшем случае (весь обмен происходит через сеть)
- Интенсивность обмена в полученном результате
- Качество решения (согласно указанному в условии)
- Время работы алгоритма для tries_count попыток анализа

Анализ алгоритма

Алгоритм работает верно на всех наборах данных, которые удалось проанализировать (алгоритм завершился). Критерием «верности» я считал соответствие распределения задач по процессорам максимальным возможным нагрузкам на них (функция bool match_condition(vector<int> &, vector<int> &, vector<int> &)), а также тот факт, что активная суммарная интенсивность обмена в сети уменьшается с каждым проходом цикла, указанного в пунктах 3-6 условия (функция int active_intensities_sum(vector<int> &, vector<vector<int>> &)).

Алгоритм не всегда завершается за допустимое время. На наборах для 8 или 16 процессоров и 100% нагрузки алгоритм ни разу не завершился. На наборах для 16 процессоров и 80% нагрузки алгоритм завершается успешно на 70-80% наборов данных (в зависимости от количества задач).

Таблица с результатами работы алгоритма находится в файле “alg_table.xlsx”.

Идеи по оптимизации алгоритма

Видно, что довольно сложная по вычислительной нагрузке O(n^2), где n – количество задач, функция int active_intensities_sum(vector<int> &, vector<vector<int>> &) работает довольно часто (преимущественно в двойных циклах, поэтому сложность возрастает до O(n^4)), и явно количество ее использований можно снизить. Для операции а из условия удалось написать функцию int active_intensities_sum_1task(int, vector<int> &, vector<int> &), которая работает за O(n) и вычисляет интенсивность взаимодействия одной задачи со всеми остальными, а не общую (для всех задач со всеми).
Возможно, также есть шанс снизить вычислительную сложность, если использовать вместо STL vector<int> обычные массивы int *.

Информация по содержимому data и генератору

В папке data находятся результаты работы генератора “gen_MihailovDM_213.exe” для указанных в их названиях параметров (количество процессоров и доля производительности). За исключением указанных все остальные параметры генератора неизменны (как в параграфе 6 условия). В папке data также находится шаблон файла параметров “gen_input.txt” для генератора (с параметрами 4 процессора, 60% доля от общей производительности), который можно непосредственно использовать (если перенести в папку src). В генераторе возможен ввод вручную, следуя выводимым на экран инструкциям, если при начале работы ввести 0. Если ввести 1, генератор будет использовать указанный файл.
Файлы из папки data за исключением “gen_input.txt” состоят из 10 сгенерированных наборов данных, идущих подряд без пропусков строк, в каждом из которых:

- 1 строка – предельные нагрузки на процессоры
- 2 строка – нагрузки задач на процессоры
- с 3 по (<кол-во задач> + 3) строки – интенсивности взаимодействия задач