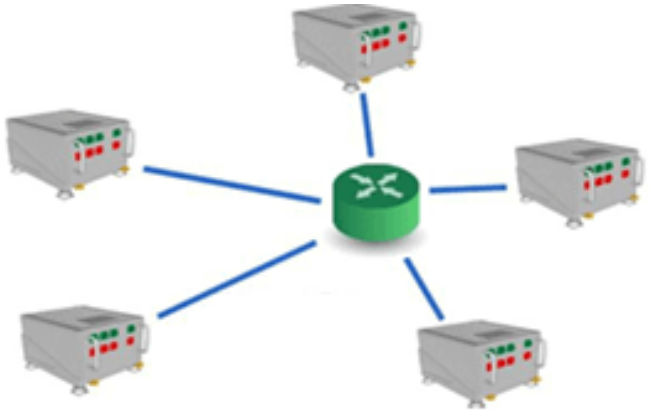
Распределение вычислительной нагрузки в многопроцессорной вычислительной системе

1 Постановка задачи

Имеется многопроцессорная вычислительная система (ВС) с одинаковыми узлами-процессорами, соединенными сетью. Задан набор программ, которые должны выполняться на этой ВС. Каждая программа выполняется на одном процессоре; допустимо выполнение на одном и том же процессоре нескольких программ. Между некоторыми программами происходит обмен данными с фиксированной скоростью. Такая архитектура используется в управляющих ВС, например, в системах авионики или автоматизации производства.



Каждая программа создает фиксированную нагрузку на процессор. Для каждого процессора задана верхняя граница допустимой нагрузки на него.

Программы, выполняющиеся на разных процессорах, обмениваются данными через сеть; при этом создается нагрузка на сеть, измеряемая в Кбайт/с. Программы, выполняющимися на одном и том же процессоре, обмениваются данными через локальную память процессора; такой обмен не создает нагрузку на сеть.

Чтобы минимизировать задержки при передаче данных через сеть, а также повысить масштабируемость системы, программы распределяют по процессорам так, чтобы нагрузка на сеть была минимальной.

Задано:

- число процессоров в ВС;
- для каждого процессора – верхняя граница нагрузки (в %, не более 100);
- число программ;
- для каждой программы: нагрузка на процессор (в %, не более 100);
- перечень пар программ, между которыми идет обмен данными;
- для каждой такой пары программ: интенсивность обмен данными (Кбайт/с).

Требуется: распределить все программы по процессорам так, чтобы минимизировать нагрузку на сеть, и при этом не нарушить ограничения по нагрузке на процессоры.

Решение задачи представляется в виде вектора $\bar{x}$ (его размер равен числу программ), в котором для каждой программы указан номер процессора, на который она распределена.

Для упрощения алгоритмов решения задачи и формирования наборов входных данных для исследования алгоритмов, на входные данные накладываются следующие дополнительные ограничения:

- верхняя граница нагрузки на процессор может принимать значения 50, 70, 90, 100 (%);
- создаваемая программой нагрузка на процессор может принимать значения 5, 10, 15, 20 (%);
- интенсивность обмена данными между программами может быть равной 0 (нет обмена), 10, 50, 70, 100 (Кбайт/с).

2 Алгоритм случайного поиска с сужением области для распределения программ по процессорам

Данный алгоритм итеративно случайным образом генерирует решение задачи $\bar{x}$ и проверяет, лучше ли оно по значению целевой функции $F(\bar{x})$ (т.е. нагрузки на сеть), чем наилучшее из ранее найденных корректных (удовлетворяющих ограничениям) решений. Если лучше, то проверяется корректность сгенерированного решения; в случае корректности, это решение становится новым наилучшим.

Схема выполнения базового алгоритма следующая:

1) рассчитать теоретическую максимальную нагрузку на сеть (для случая, когда весь обмен данными идет через сеть);
2) установить текущее наилучшее значение целевой функции (обозначим его за F_{best}) в рассчитанное на шаге 1 значение;
3) случайным образом сформировать решение задачи (вектор $\bar{x}$);
4) если $F(\bar{x}) \geq F_{best}$, перейти к шагу 3 (сформированное решение не лучше, чем ранее найденное)
5) если $\bar{x}$ – некорректное решение (нарушает ограничения по нагрузке на процессоры), перейти к шагу 3;

   // на шаге 6 имеем корректное решение, лучшее чем ранее найденное

6) обновить текущее наилучшее решение и текущее наилучшее значение целевой функции: $\bar{x}_{best} \coloneqq \bar{x}$; $F_{best} \coloneqq F(\bar{x})$;
7) перейти к шагу 3.

Завершение алгоритма происходит при нахождении корректного решения с нулевым значением F_{best}, либо когда за 1000 попыток подряд не удалось улучшить значение F_{best}, т.е. сформировать

корректное решение $\bar{x}$ такое, что $F(\bar{x}) < F_{best}$. В случае если при выполнении алгоритма не удалось найти ни одного корректного решения, результатом работы алгоритма должен быть массив, содержащий только значения (-1).

Параллельный алгоритм: алгоритм работает в виде несколько параллельно работающих «потоков», каждый из которых выполняет те же действия, что и базовый алгоритм. Потоки обновляют единое (общее для них) наилучшее решение и наилучшее значение целевой функции.

3 Формат входных данных

Набор входных данных (см. пункт «Задано» постановки задачи) задается в файле. Формат файла должен быть основан на XML. Конкретную структуру файла (элементы XML, их атрибуты) необходимо разработать.

4 Формат выходных данных

В стандартный вывод необходимо вывести:

В случае успеха:

1) строку «success»;
2) число итераций алгоритма;
3) элементы найденного решения $\bar{x}_{best}$, с разделителем-пробелом;
4) значение загрузки сети на найденном решении.

Текст по каждому из пунктов 1, 2, 3 выводить на отдельной строке.

В случае неуспеха:

1) строку «failure»;
2) число итераций алгоритма.

5 Задания и система оценки

Основное задание (обязательное, оценивается в 5 балла):

1) реализовать базовый алгоритм решения задачи в виде программы на языке C++ или Python;
2) проверить его работу на наборах данных с 4, 8, 16 процессорами; средним числом программ на процессор (число_программ / число_процессоров) не менее 8; средней загрузкой процессора не менее 50%; каждая программа ведет обмен данными не менее чем с двумя другими программами; в каждом наборе входных данных должны присутствовать все указанные в Постановке задачи варианты верхней границы нагрузки на процессор, загрузки процессора, интенсивности обмена данными;
   (наборы данных должны быть присланы вместе с программой)
3) написать инструкцию к программе:

- как собирать и запускать программу;
- описание формата входного файла.

Дополнительное задание 1 (оценивается в 2 балла):

1) реализовать параллельный алгоритм в виде многопоточной программы на том же языке, что и базовый алгоритм;
2) исследовать, на сколько (в % относительно базового алгоритма) уменьшается время выполнения программы при увеличении числа потоков (на наборах данных с 16 процессорами);
3) оформить результат исследования в виде диаграммы.

Дополнительное задание 2 (оценивается в 2 балла):

1) реализовать решение задачи в виде клиент-серверного приложения с использованием docker:
   - клиент и сервер выполняются в отдельных docker-контейнерах;
   - клиент и сервер взаимодействуют посредством XML-RPC;
   - у клиента имеется несколько комплектов входных данных;
   - клиент передает серверу комплект входных данных и ожидает ответа от сервера;
   - сервер выполняет решение задачи, передает результат клиенту и ожидает следующего комплекта входных данных;
   - после получения ответа, клиент передает серверу очередной комплект входных данных.
2) выложить docker-образы на Docker Hub (рекомендуется создать новую учетную запись и не афишировать ее);
3) написать инструкцию, как продемонстрировать решение на сайте https://labs.play-with-docker.com/

Баллы за основное и дополнительные задания суммируются. Если при проверке задания выявлен плагиат, оценка за все части задания обнуляется; проверяющие *не проводят* анализ, чье задание является «первоисточником», а чье – «заимствованием».

6 Требования к программной реализации

1) программа должна быть реализована на языке C++ или Python;
2) инструкции должны быть представлены в формате .txt или .pdf;
3) помимо технической информации, инструкция должна содержать фамилию, имя и номер группы студента;
4) программа должна работать в среде ОС Linux (проверка будет проводиться в ОС Ubuntu 18.04 или Debian Stretch);

5) текст программы должен быть хорошо читаемым, структурированным, и содержать комментарии в количестве, необходимом для его понимания;
6) программа должна диагностировать ситуации некорректного пользовательского ввода и содержимого входных файлов;
7) исходный код программы не должен выкладываться в публичные репозитории (github и т.п.)

7 Отправка работ

1) результаты выполнения заданий отправляются на адрес: asvk-nabor@asvk.cs.msu.ru
2) тема письма должна иметь вид: [номер группы] – MVS – Фамилия Имя,
   например: [211] – MVS – Винокуров Степан
3) материалы основного и дополнительных (при наличии) заданий должны быть оформлены в виде отдельных архивов формата .tar.bz2