Распределение вычислительной нагрузки в многопроцессорной вычислительной системе

1 Контекст

Имеется многопроцессорная вычислительная система (ВС) с одинаковыми процессорами и набор вычислительных задач, которые должны выполняться на этой ВС. Каждая задача выполняется на одном процессоре; допустимо выполнение на одном и том же процессоре нескольких задач. Между некоторыми задачами происходит обмен данными с фиксированной скоростью, задаваемой в Кбайт/с.

Каждая задача создает фиксированную нагрузку на процессор, эта нагрузка измеряется в процентах от производительности процессора. Для каждого процессора задана верхняя граница допустимой нагрузки на него.

Процессоры в ВС соединены друг с другом сетью, по которой идет обмен данными между задачами, выполняющимися на разных процессорах; обмен данными через сеть создает нагрузку на сеть (измеряется в Кбайт/с). Обмен данными между задачами, выполняющимися на одном и том же процессоре, выполняется через локальную память процессора и не создает нагрузку на сеть.

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

2 Формальное представление задачи

Представление входных данных:

- Число процессоров: N_P
- Верхние границы для нагрузки на процессоры (%):
  $C_1, C_2, \dots, C_{N_P}; 0 < C_i \leq 100, i = 1, \dots, N_P$
- Число задач: N_T
- Нагрузка на процессор, создаваемая задачами (%):
  $L_1, L_2, \dots, L_{N_T}; 0 < L_i \leq 100, i = 1, \dots, N_T$
- Интенсивность передачи данных между задачами (Кбайт/с), задается для пар задач:
  $P_{ij} \geq 0,\ i, j = 1, \dots, N_T$

Представление решения:

$\bar{x} = (x_1, x_2, \dots, x_{N_T}), x_i \in \overline{1, N_P},\ \ i = 1, \dots, N_T,$

x_i – номер процессора, на который распределена i-я задача.

Ограничение на корректность решения:

$$\left(\sum_{i:\, x_i = k} L_i \right) \leq C_k,\ \ k = 1, \dots, N_P$$

Минимизируемая функция (суммарная нагрузка на сеть):

$$F(\bar{x}) = \left(\sum_{\substack{i,j=1,\dots,N_T \\ i \neq j \\ x_i \neq x_j}} P_{ij} \right)$$

$F(\bar{x})$ также называют *целевой функцией*.

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

- Лимит по нагрузке на процессор (C_i) может принимать значения 40, 60, 80, 100 (%);
- Создаваемая задачей нагрузка на процессор (L_i) может принимать значения 5, 10, 20 (%).
- Интенсивность обмена данными между задачами может быть равной 0 (нет обмена), 10, 50, 100 (Кбайт/с).

3 Что требуется сделать

1. Реализовать алгоритм решения задачи (один из алгоритмов, описанных в разделе 4; какой именно – указано в сопроводительном письме).
2. Реализовать генератор наборов входных данных для алгоритма (схема генерации описана в разделе 4.2).
3. Выполнить исследование алгоритма на сгенерированных наборах входных данных по критериям: качество решения; длительность работы (методика исследования описана в разделе 6).

4 Алгоритмы распределения задач по процессорам

4.1 Жадный алгоритм

Данный алгоритм пошагово достраивает решение задачи, на каждом шагу назначая очередную задачу на тот процессор, где есть достаточный резерв производительности для ее выполнения и где находятся задачи, наиболее интенсивно (среди уже назначенных на процессоры) взаимодействующие с текущей задачей.

Схема выполнения алгоритма следующая:

1) выбрать еще не назначенную задачу *T*, которая наиболее интенсивно обменивается данными с уже назначенными на процессоры задачами (если назначенных задач нет, выбрать задачу, наиболее интенсивно обменивающуюся данными суммарно со всеми задачами)
2) среди процессоров, на которые можно назначить *T* без нарушения ограничений по нагрузке на процессор, выбрать процессор *P*, при назначении *T* на который будет максимизирован обмен

данными между *T* и прочими задачами, назначенными на этот процессор

(если назначенных задач нет, выбрать процессор с наибольшей верхней границей для нагрузки)

3) назначить *T* на процессор, выбранный на шаге 2
4) перейти к шагу 1

Алгоритм завершается, когда все задачи назначены на процессоры, либо когда на шаге 2 для очередной задачи не удалось найти процессор, на который можно назначить задачу без нарушения ограничений (в этом случае алгоритм завершается неуспешно).

4.2 Алгоритм случайного поиска с сужением области

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

Схема выполнения 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 на шаге 3 в случае превышения лимита на количество попыток случайного формирования решений (т.е. количества выполнений шага 3). Результатом работы алгоритма является текущее наилучшее решение $\bar{x}_{best}$, либо неуспех, если не удалось сформировать ни одного корректного решения.

При исследовании алгоритма рекомендуется устанавливать лимит на количество попыток случайного формирования решений в 1 000 000; если при таком значении лимита алгоритм быстро завершается (в пределах 10 секунд), увеличить лимит в 10 раз и определить, улучшается ли за счет этого качество находимого решения, т.е. уменьшается ли значение F_{best}.

Поскольку алгоритм является рандомизированным, при исследовании рекомендуется выполнять несколько его прогонов на одних и тех же входных данных и брать наихудший результат (тем самым будет получена «безопасная» оценка качества найденного решения).

4.3 Алгоритм градиентного спуска со случайным начальным приближением

Данный алгоритм случайным образом генерирует начальное приближение (корректное решение задачи) и затем итеративно улучшает его, выбирая и применяя одну из операций модификации решения. Среди возможных операций выбирается та, которая наиболее существенно улучшает значение целевой функции на решении.

Определены следующие операции:

A. Переместить задачу с одного процессора на другой (при условии, что это не приведет к нарушению ограничений по нагрузке на процессор).

B. «Поменять местами» задачи, привязанные к различным процессорам (при условии, что это не приведет к нарушению ограничений по нагрузке на процессоры). Если обе задачи создают одинаковую нагрузку на процессоры, операция B гарантированно переводит корректное решение в корректное.

C. «Поменять местами» задачу, привязанную к одному процессору, с несколькими задачами, привязанными к другому процессору, при условии, что нагрузка на процессор, создаваемая этой задачей и этой группой задач, одинаковы. Операция C гарантированно переводит корректное решение в корректное.

Схема выполнения алгоритма следующая:

1) случайным образом сформировать корректное решение задачи (если случайно сформированное решение некорректно, т.е. не удовлетворяет ограничениям по нагрузке на процессор, повторять попытки формирования решения)
2) взять сформированное на шаге 1 корректное решение в качестве текущего приближения
3) рассмотреть все возможные варианты применения операций A, B, C (по-одной), переводящие текущее приближение в (другое) корректное решение
4) выбрать вариант, наиболее существенно уменьшающий целевую функцию на текущем приближении
5) применить операцию, выбранную на шаге 4, получив в результате новое текущее приближение
6) перейти к шагу 3

Алгоритм завершается, когда на шаге 4 не удается выбрать операцию, которая бы позволила перейти к корректному решению с меньшим, чем на текущем приближении, значением целевой функции.

Поскольку алгоритм является рандомизированным, при исследовании рекомендуется выполнять несколько его прогонов на одних и тех же входных данных и брать наихудший результат (тем самым будет получена «безопасная» оценка качества найденного решения).

5 Генерация входных данных

Генератор входных данных случайным образом формирует наборы входных данных для реализованного алгоритма.

Для хранения сформированных наборов входных данных рекомендуется использовать структурированные текстовые файлы (формат на основе XML или JSON), с хранением каждого набора данных в отдельном файле.

Генератор принимает на вход следующие параметры:

- Число процессоров в системе (константа)
- Вероятности сопоставления задачам конкретных нагрузок на процессор (набор из трех целых чисел, в сумме дающих 100; задает вероятности (в %) того, что очередной задаче будет сопоставлена нагрузка, равная (соответственно) 5, 10, 20 (%))
- Вероятности сопоставления процессорам конкретных лимитов нагрузки (набор из четырех целых чисел, в сумме дающих 100; задает вероятности (в %) того, что очередному процессору задаче будет сопоставлен лимит нагрузки, равный (соответственно) 40, 60, 80, 100 (%))
- Q – доля (в %) от суммарного по всем процессорам «запаса производительности», которую должен занимать набор задач (целое число, не большее 100); например: если есть два процессора с лимитами нагрузки 40% и 80%, то «запас производительности» равен 120%; если параметр «доля суммарного запаса» равен 50 (%), то суммарная нагрузка на процессоры по всем задачам из генерируемого набора задач должна быть приблизительно равна 120*50/100 = 60 (%); это могут быть задачи с нагрузками 20, 20, 10, 5, 5
- Вероятности сопоставления парам задач конкретных интенсивностей обмена данными (набор из четырех чисел, в сумме дающих 100; задает вероятности (в %) того, что паре задач будет назначена интенсивность обмена данными, равная (соответственно) 0, 10, 50, 100

Параметры подаются на вход генератору в виде текстового файла.

Генерация одного набора входных данных выполняется по следующей схеме.

Генерация набора задач:

1) назначить процессорам лимиты нагрузки в соответствии с вероятностями назначения лимитов нагрузки
2) рассчитать суммарный по всем процессорам «запас производительности» R
3) пересчитать «запас производительности» с учётом доли, которую допустимо занимать: R:=R*Q/100
4) добавить новую задачу, назначив ей нагрузку на процессор в соответствии с вероятностями назначения нагрузок

   Примечание: при генерации входных данных информация о конкретном распределении задач по процессорам недоступна и не используется
5) если после добавления задачи суммарная (по всем задачам) нагрузка на процессоры превысила запас производительности R, отбросить задачу и запретить генерацию задач с этим значением нагрузки
6) если все значения нагрузки на процессор запрещены, **стоп**

   Примечание: если запрещены все значения нагрузки на процессор (в т.ч. наименьшее), это означает, что добавление задачи даже с наименьшей возможной нагрузкой привело к превышению запаса по производительности, и следует завершить формирование набора задач.
7) перейти к шагу 3

В соответствии с данной схемой, количество задач в наборе не задается заранее, а определяется по результатам генерации набора задач.

Генерация обменов данными:

1) для каждой пары задач: назначить этой паре интенсивность обмена данными в соответствии с вероятностями назначения интенсивностей обмена

При генерации входных данных по описанной выше схеме, задача гарантированно имеет хотя бы одно корректное решение.

6 Методика исследования алгоритма

Реализованный алгоритм следует прогнать на наборах входных данных, сформированных со следующими значениями параметров генератора:

1) Число процессоров: 4, 8, 16
2) Вероятности назначения задачам нагрузок на процессор: (25, 40, 35)
3) Вероятности назначения лимитов нагрузки на процессоры: (30, 40, 20, 10)
4) Доля (%) от суммарного «запаса производительности»: 60, 80, 100
5) Вероятности назначения парам задач интенсивностей обмена данными: (50, 15, 15, 20)

В пунктах 1) и 4) указаны несколько вариантов значений параметров генератора. Исследование должно проводиться для всех комбинаций этих вариантов.

Для каждой комбинации должны быть сгенерированы не менее 10 наборов данных. На каждом из наборов должна быть прогнана реализация алгоритма. Должны быть замерены:

- длительность работы реализации
- качество решения (отношение нагрузки на сеть на найденном решении к теоретической максимальной нагрузке на сеть, рассчитываемой для случая, когда весь обмен данными идет через сеть)

Результат прогонов должен быть отражен в таблице. Строка таблицы соответствует комбинации параметров генерации набора входных данных (например: 8 процессоров, доля запаса 60%). В столбце таблицы указывается минимальная и максимальная (по всем наборам входных данных, сгенерированных с этими параметрами) длительность прогона алгоритма, наилучшее и наихудшее качество решения (меньше – лучше).

Если для некоторых наборов данных решение не найдено (алгоритм завершился неуспешно или не завершил работу в течение 5 минут), число таких наборов также должно быть указано в таблице.

7 Требования к программной реализации и оформлению результата

В качестве языка программирования необходимо использовать C или C++. Полученная реализация должна удовлетворять следующим требованиям:

1. Код программы должен быть чистым и аккуратным. Он должен содержать комментарии в количестве, необходимом для его понимания. Подробнее про стиль кодирования можно прочитать, например, в [1], [2].
2. Архив с решением должен содержать Makefile, необходимый для сборки и запуска программы. В списке целей должны присутствовать цели *all* и *clean* (подробнее про написание и структуру Makefile'ов можно почитать, например, в [3]).
3. Архив с решением должен содержать всё необходимое для сборки и запуска программы в linux-based операционной системе (в частности, проверяться задание будет на компьютере под управлением Debian 9.0 «Stretch»). Если у вас в распоряжении имеется только компьютер с Windows, для проверки работоспособности решения на linux'е имеет смысл воспользоваться эмулятором [4] и/или виртуальной машиной.
4. Все исходные файлы (за исключением Makefile'а) должны находиться в папке *src*, включённой в архив с решением.
5. Получаемый исполняемый файл должен быть назван по шаблону:
   <префикс>_<ФамилияИО>_<номер группы>
   где <префикс> – gen для генератора и alg для реализации алгоритма
   Например, gen_IvanovAV_201.

6. Архив с решением должен содержать текстовый файл «readme.txt», содержащий:
   а) ФИО сдающего задание;
   б) номер группы;
   в) список использованных библиотек, если функционала, предоставляемого стандартной, было недостаточно.
7. Архив с решением должен содержать файл формата PDF с описанием результатов исследования, а также, в отдельной папке «data» – сгенерированные наборы входных данных, использованных для исследования, и файлы с настройками генератора наборов данных.
8. Архив с решением должен иметь формат zip и имя <ФамилияИО>_<номер группы>.zip (например, IvanovAV_201.zip). Глубина вложенности — один уровень, т.е. в архиве не должно быть отдельной корневой папки с файлами.

Литература

[1] *90 рекомендаций по стилю написания программ на C++* [HTML] (http://habrahabr.ru/post/172091/)
[2] *Google C++ Style Guide* [HTML] (https://googlestyleguide.googlecode.com/svn/trunk/cppguide.html)
[3] *Makefile для самых маленьких* [HTML] (http://habrahabr.ru/post/155201/)
[4] *Cygwin* [HTML] (www.cygwin.com)