Распределение и планирование вычислений в многоканальной многопроцессорной системе цифровой обработки сигналов

И. В. Стручков
igor@ftk.spbstu.ru

Санкт-Петербургский государственный политехнический университет

Представлена методика планирования и распределения параллельных заданий в многоканальной многопроцессорной системе цифровой обработки сигналов. Система параллельных заданий описывается графом генераторов и преобразователей данных. Алгоритм планирования является динамическим, и основан на алгоритме Earliest Deadline First (EDF). Для учета зависимостей по данным между заданиями вводится понятие смещенных периодических заданий. Распределение заданий в многопроцессорной системе осуществляется методом имитации отжига.

Введение

Цифровая обработка сигналов (ЦОС) является одной из важнейших областей применения высокопроизводительных вычислительных средств. Важными сферами применения систем ЦОС являются радиолокация и гидроакустика, где эти системы применяются для обнаружения и идентификации движущихся объектов, измерения местоположения и скорости движения объектов, их классификации. Особенностями таких систем являются большое коли-

чество параллельных каналов ввода и жесткие ограничения реального времени. Указанные особенности определяют необходимость использования вычислительных систем особого класса — многоканальных многопроцессорных систем ЦОС (ММСЦОС). Такие системы строятся на основе специализированных сигнальных процессоров, объединенных в многопроцессорные вычислительные системы с распределенной памятью.

Вычислительный процесс ММСЦОС представляет собой множество связанных по данным этапов вычислений. Важнейшим способом повышения производительности таких вычислений является параллельная обработка, которая включает в себя конвейеризацию отдельных этапов и параллелизм многоканальной обработки в рамках одного этапа. В многопроцессорной вычислительной системе параллельная обработка достигается путем назначения частей вычислений (заданий) на отдельные процессоры, которые функционируют независимо друг от друга. Взаимосвязь заданий осуществляется путем передач данных по имеющимся коммуникационным каналам многопроцессорной системы.

Распределение заданий в многопроцессорной системе должно способствовать повышению эффективности функционирования программной системы. Под ней понимается временная эффективность и эффективность использования оперативной памяти. Важнейшим требованием к разрабатываемой программной системе являются ограничения реального времени.

Таким образом, в данной статье рассматривается задача оптимального распределения заданий по вычислительным ресурсам многопроцессорной системы и планирования вычислительного процесса с целью повышения эффективности функционирования программной системы в условиях реального времени.

1. Обзор смежных работ

Для формального описания вычислительных процессов широкое применение находят модели потоков данных (Data Flow). В системах ЦОС наиболее часто применяется модель синхронных потоков данных (Synchronous Data Flow — SDF) [14]. Модель SDF представляет собой граф, каждая вершина которого может срабатывать, потребляя заданное количество элементов данных (токенов) по каждой входящей дуге и вырабатывая заданное количество токенов по каждой исходящей дуге. Управление осуществляется ис-

ключительно данными, то есть вершина срабатывает, если на всех входящих дугах имеется достаточное количество токенов.

В ряде работ исходная модель SDF расширяется с целью получения более общих результатов либо лучшего соответствия предметной области. В [9] снимается ограничение, согласно которому каждый узел при каждой итерации генерирует строго постоянное число токенов. В работе [20] рассматриваются модель масштабируемых синхронных потоков данных, в которых кратности могут изменяться в целое число раз, и модель параметрических синхронных потоков данных, в которых кратности задаются через внешние параметры. В работе [11] в модель SDF вводится дополнительный параметр — пороговый размер данных. При формализации вычислительных процессов в ММСЦОС указанные модели вычислений не позволяют адекватно описать алгоритмы с накоплением данных и параллелизм многоканальной обработки. В связи с этим в данной работе используется специально разработанная модель - граф генераторов и преобразователей данных (ГГПД) [5], расширяющая модель SDF с целью более адекватного описания вычислительных процессов в ММСЦОС.

Одной из важных особенностей модели SDF является возможность статического планирования вычислительного процесса как на одном процессоре, так и в многопроцессорной системе. Широким классом алгоритмов планирования является списковое планирование, при котором задания помещаются в сортированный список, из которого они поочередно изымаются и назначаются на свободный процессор. Алгоритмы различаются в зависимости от порядка сортировки заданий и выбора свободного процессора: Modified Critical Path (MCP), Mobility-Directed (MD) [21], Highest Levels First with Estimated Times (HLFET), Highest Dynamic Levels First with Estimated Times (HDLFET), Dynamic Level Scheduling (DLS) [16]. В частности, алгоритм DLS реализован в среде моделирования и разработки встраиваемых систем Ptolemy*. Альтернативным подходом является кластеризация графа заданий [17, 19]. Алгоритмы спискового планирования и кластеризации позволяют оптимизировать время выполнения набора связанных заданий, но показывают невысокие результаты при планировании многократно повторяющихся заданий, так как не учитывают параллелизм между итерациями.

* http://ptolemy.eecs.berkeley.edu.

Различают циклические и периодические повторяющиеся задания. Циклические задания связаны с наличием замкнутых контуров в графе заданий. Периодически повторяющиеся задания связаны с периодическим поступлением данных на вход системы, что характерно для систем ММСЦОС [4]. Периодические задания требуют динамического планирования, то есть принятия решения о порядке выполнения заданий по факту прихода запросов в соответствии с системой приоритетов. При этом в системах реального времени могут накладываться ограничения на время обработки каждого запроса (директивные сроки выполнения). В фундаментальной работе [15] доказывается оптимальность алгоритмов Rate Monotonic (RM) со статическими приоритетами и Earliest Deadline First (EDF) с динамическими приоритетами для планирования периодических независимых заданий реального времени. В [10] предложен способ модификации алгоритма EDF для заданий с зависимостями по данным для бесконтурных графов заданий, в [23] данный метод расширяется на случай графа с контурами. Однако оба указанных метода не допускают возможности прерывания задания более приоритетным (вытесняющей многозадачности). В [7] рассматривается алгоритм EDF для систем спорадических заданий, являющихся более общим случаем по сравнению с периодическими.

При планировании заданий в многопроцессорной системе с распределенной памятью необходимо учитывать коммуникационные затраты на передачу данных между процессорами. В [18] подробно изучается влияние коммуникационных затрат и предлагаются модификации спискового алгоритма планирования с учетом этих затрат. В работах [6,8] рассматриваются случаи гетерогенных вычислительных систем с распределенной памятью, когда процессоры и коммуникационные каналы имеют различные характеристики. Однако данные работы не учитывают возможности периодического выполнения заданий и требования реального времени.

Из приведенного обзора следует, что для заданий в системах ММСЦОС целесообразно применять динамическое планирование с учетом зависимостей заданий по данным и коммуникационных затрат. Эти зависимости приводят к тому, что, несмотря на периодическое поступление входных данных, задания на отдельных процессорах системы не являются ни периодическими, ни спорадическими. В данной работе вводится понятие смещенных периодических заданий и доказывается применимость алгоритма планирования EDF для этого типа заданий.

Предлагаемый подход разделяет этапы планирования вычислений и распределения заданий по процессорам. Для оптимального распределения заданий могут применяться различные алгоритмы, одним из наиболее точных среди которых является алгоритм имитации отжига [12, 22]. При этом, в отличие от подхода, описанного в [12], в данной работе критерием оптимизации является не балансировка загрузки, а непосредственно показатели эффективности программной системы для ММСЦОС: время распространения данных, пропускная способность или размеры буферов передачи.

2. Модель вычислительного процесса

Модель вычислительного процесса в виде графа генераторов и преобразователей данных (ГГПД) [5] включает три типа элементарных функциональных узлов: генератор, преобразователь и терминатор. При реализации вычислительного процесса на многопроцессорной системе данные функциональные узлы соответствуют элементарным частям исполняемой программы — заданиям. Генератор периодически формирует всплески данных с заданным периодом и характеризуется следующими параметрами: периодом всплесков (T_B), количеством фрагментов данных во всплеске (N_B), множеством получателей данных ($\mathcal{R}$). Преобразователь данных, в отличие от генератора, активизируется не автономно, а после получения заданного количества входных данных. Преобразователь характеризуется следующими параметрами: количеством входных фрагментов данных для активизации (N_A), временем обработки (T_O), множеством получателей данных ($\mathcal{R}$), для каждого получателя важны следующие параметры: количество фрагментов данных во всплеске (N_B) и делитель, то есть количество активизаций, после которого генерируется всплеск (M). Делитель преобразователя позволяет моделировать вычисления с накоплением данных. Терминатор не имеет параметров и обозначает конец вычислений.

В дополнение к указанным характеристикам модель ГГПД имеет следующие дополнительные параметры заданий, используемые при планировании и распределении вычислений: тип расщепления данных [3], переместимость и расщепляемость задания, объем требуемой оперативной памяти для задания.

3. Расщепление и слияние заданий

Расщепление заданий осуществляется с целью использования параллелизма, скрытого в алгоритме конкретного задания, а также для оптимизации потоков данных в вычислительной системе. При расщеплении создается несколько экземпляров одного задания, каждый из которых выполняет часть вычислений. Все экземпляры заданий могут выполняться параллельно. Процесс, обратный расщеплению, называется слиянием заданий.

Использование понятия расщепления заданий позволяет ввести в рассмотрение особый класс графов ГГПД, часто встречающийся на практике, — ГГПД-дерево с расщеплением. Таким деревом будем называть бесконтурный граф ГГПД, полученный из ориентированного дерева путем расщепления отдельных вершин с копированием инцидентных им дуг.

4. Расчет интенсивностей потоков данных, загрузки процессоров и коммуникационных каналов

Средней интенсивностью потока данных от генератора или преобразователя называется отношение величины всплеска потока к периоду всплесков:

$$\lambda_{ij} = \frac{N_{B\,ij}}{T_{B\,ij}}, \tag{1}$$

где i - индекс задания-источника данных, j - индекс задания-приемника данных. Для генераторов величины $N_{B\,ij}$ и $T_{B\,ij}$ каждого генерируемого потока данных заданы в модели ГГПД. Для потоков данных, генерируемых преобразователями, величина N_{Bij} задана в модели ГГПД, а период потока вычисляется на основе периода активизации задания с учетом делителя. Период преобразователя (T_i) в модели ГГПД определяется интенсивностью входных потоков данных и числом активизации задания.

$$T_{B\,ij} = M_{ij}T_i = \frac{M_{ij}N_{Ai}}{\sum_{k\in\mathcal{I}(i)}\lambda_{ki}}, i \in \mathcal{X}, \tag{2}$$

где $\mathcal{I}(i)$ — множество заданий-источников для задания i, $\mathcal{X}$ – множество всех преобразователей, N_{Ai} – количество данных для активизации i-го преобразователя, M_{ij} – делитель j-го потока преобразователя i.

При расщеплении заданий важным вопросом является перераспределение потоков данных между экземплярами заданий. В [3] приведен алгоритм расчета интенсивностей потоков данных между экземплярами расщепленных заданий с учетом типа расщепления данных.

Коэффициент загрузки процессора определяется согласно формуле.

$$U_p = \sum_{i \in \mathcal{X}(p)} \frac{T_{Oi}}{T_i}, \tag{3}$$

где $\mathcal{X}(p)$ — множество преобразователей, размещенных на процессоре p.

Коэффициент загрузки коммуникационного канала определяется, как отношение суммарной интенсивности передаваемых по каналу данных к пропускной способности канала (4). Состав потоков данных для некоторого канала l определяется согласно функции маршрутизации $RA(i,j,l) : RA(i,j,l) = 1$ тогда и только тогда, когда поток данных между заданиями i и j проходит через коммуникационный канал l при заданном распределении заданий. Величина w_l (время передачи единицы данных по каналу) является обратной величиной к пропускной способности канала:

$$U_l = \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_{ij} w_l RA(i,j,l), \tag{4}$$

где m — количество заданий в системе.

5. Алгоритм планирования с динамическими приоритетами для смещенных периодических операций

Рассмотрим систему параллельных заданий с зависимостями по данным, заданную в виде ГГПД-дерева, и некоторое распределение заданий по процессорам в многопроцессорной системе. В [2] показано, что если для каждого задания и операции передачи данных выделен входной буфер размером на 2 запроса, то для всех заданий и операций передачи данных можно обеспечить поступление запросов с постоянным периодом путем искусственной задержки запросов в буфере на время, равное периоду источника соответствующего запроса. Соответствующее расписание выполнения заданий

и передач данных в дальнейшем именуется как периодическое многопроцессорное расписание реального времени с искусственными задержками.

Решение задачи составления расписания реального времени заключается в определении моментов старта каждой операции (t_{si}) на каждом исполнительном устройстве (процессоре или канале передачи данных). Поскольку расписание является периодическим, то достаточно определить начальные сдвиги каждой операции относительно момента старта системы, время каждого последующего запуска операции определяется известным периодом. Начальные сдвиги операций вычисляются путем суммирования сроков выполнения операций по всему пути следования данных от генератора до заданной операции. Необходимо отметить, что расписание не является статическим. Начальный момент старта операции определяет лишь момент, когда операция поступает на выполнение. Далее порядок выполнения операций определяется диспетчером с абсолютными приоритетами. При удовлетворении определенным ограничениям на загрузку исполнительного устройства могут быть гарантированы необходимые директивные сроки выполнения периодических операций при использовании как статических, так и динамических приоритетов [15].

Идея, положенная в основу представляемого подхода, состоит в том, что при использовании алгоритма планирования операций с динамическими приоритетами искусственные задержки операций могут не использоваться явно. Это следует из свойства алгоритма планирования на основе сроков выполнения (EDF), сформулированного ниже в теореме. При этом искусственные задержки используются только для расчета директивных сроков выполнения заданий, на основе которых рассчитываются приоритеты согласно алгоритму EDF. Отказ от искусственных задержек приводит к тому, что запросы к операциям не являются строго периодическими. Запрос может прийти раньше планового времени старта, однако не может прийти позже при условии, что для операции-источника справедливы директивные сроки выполнения.

Определение 1. *Смещенными периодическими операциями называются операции, для которых существуют плановые периодические точки старта с детерминированным периодом, однако запрос к операции может прийти раньше планового времени старта (но не позже).*

Теорема 1. *Алгоритм планирования смещенных периодических операций на некотором исполнительном устройстве (ИУ) на основе сроков выполнения (Earliest Deadline First — EDF) сохраняет директивные сроки выполнения операций, рассчитанные на основе периодического расписания с искусственными задержками, при условии выполнимости расписания с искусственными задержками для данного набора операций.*

Доказательство. Рассмотрим k-й период смещенной периодической операции i на некотором исполнительном устройстве, ограниченный моментами времени t^* и $t^* + T_i$, где Ti – период операции i, $t^* = t_{si} + (k-1)T_i$, t_{si} – плановый начальный момент старта операции. Поскольку точки старта смещенных периодических операций могут быть сдвинуты только влево по оси времени относительно плановых значений, то суммарное количество полезной работы, выполненной ИУ за время от старта системы до t^*, не меньше, чем в случае расписания с искусственными задержками. В самом деле, сдвиг операций влево по временной оси может привести лишь к более плотной упаковке операций во времени, что ведет к уменьшению времени простоя ИУ и, следовательно, увеличению объема выполненной работы. С другой стороны, сдвиг влево времени начала операции, плановый срок окончания которого превосходит $t^* + T_i$, не может повлиять на операцию i, так как эти операции окажутся менее приоритетными согласно алгоритму EDF. Таким образом, доля рабочего времени ИУ, которое может быть выделено операции i в течение рассматриваемого периода, не меньше, чем в случае расписания с искусственными задержками. Следовательно, операция будет выполнена к директивному сроку, так как периодическое расписание с искусственными задержками по условию теоремы является выполнимым. □

Для расчета плановых сроков выполнения операций, являющихся основой алгоритма EDF, для системы заданий с зависимостями по данным необходимы моменты старта операций из расписания с искусственными задержками, которые не превышают период входящего потока данных (то есть период операции-источника этого потока данных). Однако искусственная задержка в размере целого периода источника данных в большинстве случаев окажется избыточной, поскольку такая задержка исходит из предположения, что наихудшее время выполнения операции соответствует предель-

ному сроку выполнения. Реальные системы имеют определенный запас производительности, в результате чего наихудшее время выполнения операции определяется фактическим сроком выполнения операции (S_i).

Определение 2. *Фактический срок выполнения смещенной периодической операции есть промежуток времени от планового момента начала текущего периода операции до полного окончания операции в наихудшем случае.*

Очевидно, что, если в системе имеется выполнимое расписание реального времени, то фактический срок выполнения операции не превышает предельный директивный срок.

Рассмотрим исполнительное устройство, на котором выполняются m операций. Пусть периоды операций упорядочены следующим образом: $T_1 < T_2 < ... < T_m$. Фактический срок выполнения операции i складывается из времени выполнения операции i и времен выполнения конкурирующих операций с более высокими динамическими приоритетами. В [4] показано, что фактический срок выполнения операции для алгоритма EDF может быть рассчитан по формуле

$$S_i = P_i + \sum_{j=1}^{i-1} S'_{ij} + S''_{i(i+1)}, \tag{5}$$

где P_i — время выполнения операции на выделенном процессоре, S'_{ij} — задержки от операций с меньшим периодом (6), $S''_{i(i+1)}$ — задержка от соседней операции с большим периодом (7):

$$S'_{ij} = \left\lfloor \frac{T_i}{T_j} \right\rfloor P_j + min\left\{ \left(T_i - T_j \left\lfloor \frac{T_i}{T_j} \right\rfloor \right), P_j \right\}; \tag{6}$$

$$S''_{ik} = max\{0, T_i - (T_k - (S_k - P_i))\}. \tag{7}$$

Формула (5) может быть применена последовательно ко всем операциям на данном исполнительном устройстве, начиная с операции, имеющей наибольший период, и далее в порядке убывания периода. Рассчитанные по формуле (5) фактические сроки окончания операций могут быть использованы в качестве оценки длительности операций в наихудшем случае, при расчете расписания с искусственными задержками.

6. Расчет размеров буферов данных

Рассмотрим сначала буферы передачи данных, находящиеся на входе преобразователя данных. Согласно доказанной выше теореме, самый поздний срок начала обработки запроса преобразователем и максимальный срок завершения обработки запроса определяются расписанием с искусственными задержками (время старта операции — t_{si} и фактический срок выполнения операции — S_i). Самый поздний момент удаления запроса из входного буфера преобразователя i определяется выражением

$$L_i = t_{si} + S_i. \tag{8}$$

Самый ранний момент поступления запроса во входной буфер $k = 1..k_i$ преобразователя i определяется выражением

$$E_{ik} = E_j + P_j, j = src(i,k), \tag{9}$$

где $src(i,k)$ — индекс операции-источника для буфера k преобразователя i, E_j — самый ранний момент поступления объединенного запроса к операции-источнику, P_j — время выполнения операции-источника, если источник является преобразователем или операцией передачи данных, или выражением

$$E_{ik} = T_{Bj}, j = src(i,k), \tag{10}$$

если операция-источник является генератором, где T_{Bj} — период генератора.

Самый ранний момент поступления объединенного запроса на вход преобразователя i при наличии нескольких входных потоков определяется выражением

$$E_j = \max_{k=1..K_i} E_{ik}. \tag{11}$$

Максимальное количество данных, которое может быть накоплено во входном буфере задания, определяется промежутком времени между самым ранним моментом поступления и самым поздним моментом удаления данных, а также параметрами потока данных:

$$B_{ik} = \left\lceil \frac{L_i - E_{ik}}{T_{ik}} \right\rceil N_{ik}, \tag{12}$$

где T_{ik} – период потока данных, входящего в буфер k преобразователя i, N_{ik} — размер всплеска потока данных, входящего в буфер k преобразователя i.

Размер входного буфера к операции передачи данных, находящейся непосредственно после генератора или преобразователя, может быть также рассчитан по формуле (12). Особым случаем является операция передачи данных, непосредственным предшественником которой в пути следования данных является также операция передачи. Данная ситуация возникает на промежуточном узле, где не выполняется обработка данных, а только маршрутизация из одного канала в другой. В данном случае нет необходимости дожидаться полного поступления всего запроса, и операции приема и передачи могут быть частично наложены во времени.

Формализуя все вышесказанное, самый ранний момент поступления запроса во входной буфер промежуточной операции передачи данных определяется выражением

$$E'_i = E'_j + \delta_{ij}, j = src(i), \tag{13}$$

где δ_{ij} — время инициации передачи данных между операциями i и j, $src(i)$ — операция-источник данных для операции j.

Самый поздний момент начала передачи данных для промежуточной операции определяется выражением

$$L'_i = t_{si} + S_i - P_i. \tag{14}$$

Максимальный размер входного буфера данных для промежуточной операции передачи данных определяется выражением

$$B'_i = \left\lfloor \frac{L'_i - E'_i}{T_i} \right\rfloor N_i + min\left\{ \left(L'_i - E'_i - \left\lfloor \frac{L'_i - E'_i}{T_i} \right\rfloor T_i \right) C, N_i \right\}, \tag{15}$$

где N_i — размер всплеска входящего потока операции i, T_i — период входящего потока операции i, C — скорость передачи данных.

7. Распределение параллельных заданий в многопроцессорной вычислительной системе

Приведенный выше способ планирования заданий в многопроцессорной системе предполагает, что дано некоторое распределение заданий по процессорам. На практике распределение заданий по

процессорам является решением задачи оптимизации. Исходными данными для оптимизации являются: формализованное описание системы параллельных заданий в виде ГГПД, формализованное описание аппаратной системы в виде графа соединений процессоров, таблица маршрутизации данных для каждого процессора.

7.1. Использование метода имитации отжига

Согласно ряду исследований [12, 22], алгоритм имитации отжига является одним из наиболее точных известных эвристических алгоритмов решения вычислительно сложных задач оптимизации. Метод имитации отжига служит для поиска глобального минимума некоторой функции $f(x)$, заданной для x из некоторого пространства S. Элементы множества S представляют собой состояния воображаемой физической системы, а значение функции f в этих точках является аналогом энергии системы $E = f(x)$. В каждый момент предполагается заданной температура системы T, уменьшающаяся с течением времени. После попадания в состояние x при температуре T следующее состояние системы выбирается в соответствии с заданным порождающим семейством вероятностных распределений. В качестве вероятности принятия нового состояния чаще всего принимается величина $h(\Delta E, T) = exp(-\Delta E/T)$, где ΔE — приращение энергии.

Алгоритм сверхбыстрого отжига [13] обеспечивает быстрое уменьшение температуры при сохранении статистической гарантии сходимости. Данный алгоритм особенно эффективен для задач большой размерности. Каждая компонента нового вектора состояния рассчитывается по формуле $x'_i = x_i + y_i(B_i - A_i)$, где B_i и A_i — границы диапазона возможных изменений координаты. Вводится порождающая функция

$$g_T(y) = \prod_{i=1}^{D} \frac{1}{2(|y_i| + T_i)ln(1 + 1/T_i)} \ . \tag{16}$$

Каждая компонента случайного вектора y может быть смоделирована следующим образом:

$$y_i = sgn(u_i - \frac{1}{2})T_i[(1 + 1/T_i)^{|2u_i - 1|} - 1], \tag{17}$$

где u_i — случайная величина, равномерно распределенная в диапазоне $[0, 1]$.

В [13] доказано, что при изменении температуры, согласно следующему закону:

$$T_i(k) = T_{0i} exp(-c_i k^{1/D}) \ , \tag{18}$$

выполняется статистическая гарантия сходимости алгоритма.

Существенным вопросом является способ формирования вектора состояния системы x. Для того чтобы учесть возможность расщепления/слияния заданий, предлагается использовать следующий подход. Каждый диапазон значений координаты вектора состояний между ближайшими целыми числами $[a_j, b_j]$ разбивается на три зоны. При попадании значения координаты в диапазон $[a_j, a_j + 0,25[$ считается, что задание распределено на процессор a_j. При попадании значения координаты в диапазон $]b_j - 0,25, b_j]$ считается, что задание распределено на процессор b_j. При попадании значения координаты в диапазон $[a_j + 0,25, b_j - 0,25]$ считается, что данный экземпляр задания отсутствует. При этом размерность задачи увеличивается, учитывая максимальные степени расщепления всех заданий. Например, если все задания имеют одинаковую максимальную степень расщепления r, а число заданий m, то размерность задачи равна $D = rm$.

7.2. Расчет значения целевой функции оптимизации

Основной целью оптимизации распределения заданий является повышение эффективности программного обеспечения ММСЦОС. Согласно ISO 9126, понятие эффективности программного обеспечения включает временную эффективность и эффективность использования оперативной памяти [1]. Указанные цели оптимизации определяют возможные альтернативы целевой функции.

7.2.1. Оптимизация длительности обработки

Длительность пути обработки данных L вычисляется как максимальная сумма фактических сроков выполнения операций на всех исполнительных устройствах в пути передачи данных от некоторого генератора до некоторого терминатора:

$$L(g_i, t_j) = \max_{P_m \in \mathcal{P}(g_i, t_j)} \sum_{k \in P_m} S_k, \tag{19}$$

где g_i — некоторый генератор, t_j — некоторый терминатор, $\mathcal{P}(g_i, t_j)$ — множество всех путей передачи данных от g_i до t_i, S_k — фактический срок выполнения k-й операции, входящей в путь P_m из множества путей. Длительность критического пути обработки данных $L_{\max}$ тогда определяется следующим выражением:

$$L_{\max} = \max_{g_i \in \mathcal{G}, t_j \in \mathcal{T}} L(g_i, t_j), \tag{20}$$

где $\mathcal{G}$ — множество всех генераторов ГГПД, $\mathcal{T}$ — множество всех терминаторов ГГПД.

7.2.2. Оптимизация пропускной способности

Минимально возможный период поступления данных для каждого периодического задания определяется фактическим сроком выполнения этого задания (S_i). Следует также учесть, что периоды заданий в ГГПД могут различаться. Для унификации заданий целесообразно использовать относительную величину — отношение фактического срока выполнения к базисному периоду задания T_i. Базисный период задания рассчитывается в соответствии с (2) при некотором эталонном входном потоке данных. Таким образом, отношение фактического срока выполнения к базисному периоду задания определяет долю эталонного потока от максимальной пропускной способности данного задания. На пропускную способность системы влияют также операции передачи данных. Для операций передачи данных также рассчитываются фактический срок выполнения и базисный период, равный периоду всплесков соответствующего потока данных. В результате целевая функция определяется выражением

$$T_{\max} = \max_{i \in \mathcal{X} \cup \mathcal{C}} \frac{S_i}{T_i}, \tag{21}$$

где $\mathcal{X}$ — множество заданий-преобразователей, $\mathcal{C}$ — множество операций передачи данных, S_i — фактический срок выполнения операции, T_i — базисный период операции. Минимизируя величину T_{max}, находим распределенение заданий, при котором возможно максимальное увеличение интенсивности входного потока, а следовательно, максимальная пропускная способность системы в целом.

7.2.3. Оптимизация использования оперативной памяти

Наиболее существенные затраты оперативной памяти необходимы для промежуточных буферов передачи данных и хранимых данных заданий. Суммарные затраты на размещение промежуточных буферов передачи на процессоре (B_p) складываются из размеров входных буферов преобразователей (B_{ik}), размещенных на данном процессоре, и буферов операций передачи данных (B'_i), размещенных на коммуникационных каналах данного процессора. Суммарные затраты на хранимые данные заданий процессора (A_p) складываются из потребностей в оперативной памяти преобразователей, размещенных на данном процессоре (согласно параметрам, указанным в ГГПД). Суммарный объем необходимой оперативной памяти на плате (V_b) складывается из затрат оперативной памяти всех процессоров платы:

$$V_b = \sum_{p \in \mathcal{P}(b)} (A_p + B_p), \tag{22}$$

где $\mathcal{P}(b)$ — множество процессоров на плате b.

Тогда средний необходимый объем оперативной памяти на плату вычисляется следующим образом:

$$V_{cp} = \frac{1}{N} \sum_{b=1}^{N} V_b, \tag{23}$$

где N — число плат в системе.

7.3. Расчет ограничений оптимизации

Алгоритм оптимизации по методу имитации отжига характеризуется наличием одной оптимизируемой функции общего вида. В таких условиях наиболее целесообразным способом учета ограничений оптимизации является введение штрафных функций. Рассмотрим различные виды штрафных функций, использование которых определяется выбранной целью оптимизации и системными требованиями.

7.3.1. Ограничения производительности процессоров и пропускной способности коммуникационных каналов

Значение штрафной функции вычисляется по формуле

$$P_U(p|l) = \begin{cases} 0, & U_{p|l} \leq 1, \\ B_U(U_{p|l} - 1), & U_{p|l} > 1, \end{cases} \tag{24}$$

где $U_{p|l}$ — коэффициент загрузки процессора p или коммуникационного канала l, B_U — барьерное значение штрафной функции.

Окончательное значение штрафной функции определяется как сумма штрафов по каждому процессору и коммуникационному каналу в системе.

7.3.2. Ограничения на длительность обработки

Штрафная функция длительности обработки рассчитывается на основе показателя $L_{\max}$ из п. 7.2.1. При условии, что задано предельное значение длительности обработки L_{lim}, штрафная функция имеет следующий вид:

$$P_L = \begin{cases} 0, & L_{\max} \leq L_{lim}, \\ B_L(L_{\max} - L_{lim}), & L_{\max} > L_{lim}, \end{cases} \tag{25}$$

где B_L — барьерное значение штрафной функции.

7.3.3. Ограничения на размер оперативной памяти

Согласно п. 7.2.3, суммарные затраты оперативной памяти для платы V_b могут быть рассчитаны по формуле (22). При условии, что задано предельное значение доступной оперативной памяти платы V_{blim}, штрафная функция имеет следующий вид:

$$P_M(b) = \begin{cases} 0, & V_b \leq V_{blim}, \\ B_M(V_b - V_{blim}), & V_b > V_{blim}. \end{cases} \tag{26}$$

Окончательное значение штрафной функции определяется как сумма штрафов по каждой плате в системе.

8. Экспериментальные исследования

Для того чтобы оценить преимущества использования представленного в данной статье подхода при проектировании программного обеспечения ММСЦОС были проведены эксперименты по определению эффективности планирования заданий в многопроцессорной системе. Поскольку множество возможных графов заданий бес-

конечно, в данных экспериментах выбиралось определенное количество случайно сгенерированных графов заданий в виде ГГПД-дерева с расщеплением. Выбор дерева с расщеплением в качестве базового класса обоснован тем, что, во-первых, приведенные в данной работе формулы и алгоритмы справедливы именно для этого класса (хотя могут быть обобщены), во-вторых, данный класс графов заданий весьма часто встречается в системах ММСЦОС. В качестве топологии многопроцессорной системы был выбран 8-процессорный полносвязный граф. Всего в рамках каждого эксперимента было выполнено по 100 реализаций, что позволило обеспечить приемлемые доверительные интервалы результатов при доверительной вероятности 0,95.

В первой серии экспериментов проводилось сравнение длительности одной итерации в расписании, полученном с помощью представленной методики, с результатом применения алгоритма DLS. При этом была использована реализация алгоритма DLS в системе Ptolemy. Для каждого эксперимента был сгенерирован ГГПД, состоящий из 23 заданий. Времена выполнения заданий-преобразователей выбирались случайно согласно равномерному распределению вероятностей из диапазона [4, 40]. Размеры данных, формируемых и потребляемых заданиями за одну активизацию, выбирались также согласно равномерному распределению из диапазона [5 000 000, 50 000 000] слов при условии, что одно слово передается по коммуникационному каналу в течение $2 \cdot 10^{-7}$ единиц времени. При этом числа активизации заданий-приемников выбирались равными размерам всплесков заданий-источников. Таким образом, построенные ГГПД соответствуют одночастотным графам SDF. В каждой реализации эксперимента оценивалось отношение длительности итерации, полученной с помощью алгоритма DLS, к соответствующей величине, полученной исследуемым способом. По результатам первого эксперимента исследуемый способ планирования на $(13 \pm 3)\%$ превосходит алгоритм DLS.

Во втором эксперименте при тех же условиях исследовалась другая характеристика — пропускная способность. По данной характеристике исследуемый способ планирования превосходит алгоритм DLS на $(198 \pm 11)\%$. Такое значительное преимущество объясняется тем, что алгоритм DLS рассматривает только одну итерацию вычислений и не может учитывать наложение итераций.

Чтобы поставить алгоритм DLS в более выгодные условия, в третьем эксперименте исследовались многочастотные графы зада-

ний, сгенерированные аналогично первому эксперименту, но с условием, что отношение числа активизации преобразователя к размеру всплеска входящего потока может быть 2, 3, 5 или 7. Такой граф задания эквивалентен многочастотному графу SDF. При планировании многочастотных графов SDF по алгоритму DLS выполняется предварительное преобразование многочастотного графа к развернутому одночастотному графу APG (Acyclic Precedence Graph). Благодаря этому, алгоритм DLS может анализировать одновременно несколько итераций, улучшая тем самым пропускную способность. Исследуемый алгоритм для многочастотного графа работает так же, как и в случае одночастотного. По результатам планирования многочастотных графов показатель пропускной способности исследуемого алгоритма на $(179 \pm 10)\%$ выше. Таким образом, во всех рассмотренных случаях исследуемый способ позволяет повысить такие характеристики эффективности разрабатываемых систем, как длительность обработки и пропускная способность, по сравнению с алгоритмом DLS.

Заключение

В данной статье представлена методика планирования и распределения параллельных заданий в системах ММСЦОС. Вычислительный процесс формализуется с помощью модели графа генераторов и преобразователей данных. Для планирования заданий в многопроцессорной системе может быть использован алгоритм EDF для смещенных периодических заданий. Использование данного алгоритма позволяет учесть периодичность заданий и коммуникационные затраты, рассчитать необходимые размеры буферов передачи. Этап распределения заданий отделен от этапа планирования и выполняется с помощью алгоритма сверхбыстрого отжига. Целевая функция оптимизации распределения заданий позволяет оптимизировать длительность обработки, средние размеры буферов данных или пропускную способность. Экспериментальное исследование эффективности представленного способа планирования в сравнении с известным алгоритмом DLS свидетельствует о существенном выигрыше от использования представленного способа.

Список литературы

[1] *Липаев В. В.* Методы обеспечения качества крупномасштабных программных средств. М.: СИНТЕГ, 2003. 520 с.

[2] *Стручков И. В.* Двухэтапный метод распределения периодических задач реального времени в многопроцессорной системе // Информационные технологии моделирования и управления. Вып. 6(31). Воронеж, 2006. С. 744-753.

[3] *Стручков И. В.* Распределение вычислений с расщеплением задач в многопроцессорной системе обработки сигналов // Процессы управления и устойчивость: Труды 38-й международной научной конференции аспирантов и студентов. Россия, СПб., 9–12 апреля 2007 г. / Под. ред. А. В. Платонова, Н. В. Смирнова. СПб.: Изд-во СПбГУ, 2007. С. 457–462.

[4] *Стручков И. В.* Распределение и планирование периодических задач и потоков данных в многопроцессорной вычислительной системе // Системы управления и информационные технологии. Вып. 4.2(26). Воронеж, 2006. С. 271–276.

[5] *Стручков И. В., Ицыксон В. М.* Формализм для описания программных систем и вычислительных процессов циклической параллельной обработки данных реального времени // Информационно-управляющие системы. Вып. 2. СПб., 2006. С. 8–13.

[6] *Amoroso A., Marzullo K.* Multiple Job Scheduling in a Connection-Limited Data Parallel System // IEEE Transactions on Parallel and Distributed Systems. Vol. 17. N 2. February 2006. P. 125–134.

[7] *Baruah S., Fisher N.* The Partitioned Multiprocessor Scheduling of Deadline-Constrained Sporadic Task Systems // IEEE Transactions on Computers. Vol. 55. N 7. July 2006. P. 918–923.

[8] *Berten V., Goossens J., Jeannot E.* On the Distribution of Sequential Jobs in Random Brokering for Heterogeneous Computational Grids // IEEE Transactions on Parallel and Distributed Systems. Vol. 17. N 2. February 2006. P. 113–124.

[9] *Bilsen G., Engels M., Lauwereins R., Peperstraete J. A.* Cyclo-static data flow // International Conference on Acoustics, Speech, and Signal Processing, 1995 (ICASSP-95). Vol. 5. 9–12 May 1995. P. 3255–3258.

[10] *Blazewicz J.* Scheduling Dependent Tasks with Different Arrival Times to Meet Deadlines // Modeling and Performance Evaluation of Computer Systems. Amsterdam: North-Holland, 1976. P. 57–65.

[11] *Goddard S., Jeffay K.* Managing Memory Requirements in the Synthesis of Real-Time Systems from Processing Graphs // Proceedings of the Fourth IEEE Real-Time Technology and Applications Symposium, 1998. P. 59–70.

[12] *Hluchý L., Dobrovodský M., Dobrucký M.* Static Mapping Methods for Processor Networks. http://citeseer.ist.psu.edu/472693.html.

[13] *Ingber L.* Very fast simulated re-annealing // Mathematical and Computer Modelling. Vol. 12(8). 1989. P. 967–973.
http://citeseer.ist.psu.edu/article/ingber89very.html.

[14] *Lee E. A., Messerschmitt D. G.* Synchronous Data Flow // Proceedings of the IEEE. Vol. 75. N 9. September 1987. P. 1235–1245.

[15] *Liu C., Layland J.* Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment // Journal of the ACM. Vol. 20(1). January 1973. P. 46–61. http://citeseer.ist.psu.edu/liu73scheduling.html.

[16] *Sih G. C., Lee E. A.* A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures // IEEE Transactions on Parallel and Distributed Systems. Vol. 4. N 2. February 1993. P. 175–187.

[17] *Sih G. C., Lee E. A.* Declustering: A New Multiprocessor Scheduling Technique // IEEE Transactions on Parallel and Distributed Systems. Vol. 4. N 6. June 1993. P. 625–637.

[18] *Sinnen O., Sousa L. A., Sandnes F. E.* Toward a Realistic Task Scheduling Model // IEEE Transactions on Parallel and Distributed Systems. Vol. 17. N 3. March 2006. P. 263–275.

[19] *Tao Yang, Gerasoulis A.* DSC: scheduling parallel tasks on an unbounded number of processors // IEEE Transactions on Parallel and Distributed Systems, September 1994. Vol. 5. N 9. P. 951–967.

[20] *Theelen B. D., Geilen M. C. W., Basten T. et al.* A scenario-aware data flow model for combined long-run average and worst-case performance analysis // Proceedings of the Fourth ACM and IEEE International Conference on Formal Methods and Models for Co-Design, 2006 (MEMOCODE'06). 27–30 July 2006. P. 185–194.

[21] *Wu M.-Y., Gajski D. D.* Hypertool: a programming aid for message-passing systems // IEEE Transactions on Parallel and Distributed Systems. July 1990. Vol. 1. N 3. P. 330–343.

[22] *Yarkhan A., Dongarra J.* Experiments with Scheduling Using Simulated Annealing in a Grid Environment // M. Parashar, editor, Lecture notes in computer science 2536 Grid Computing — GRID 2002 International Workshop. Vol. 3. Baltimore, MD, USA, November 2002. Springer Verlag. P. 232–242.
http://citeseer.ist.psu.edu/yarkhan02experiments.html.

[23] *Ziegenbein D., Uerpmann J., Ernst R.* Dynamic response time optimization for SDF graphs // IEEE/ACM International Conference on Computer Aided Design, 2000 (ICCAD-2000). Vol. 5–9. November 2000. P. 135–140.