На правах рукописи

НЕДОВОДЕЕВ Константин Владимирович

МЕТОДЫ ПОСТРОЕНИЯ ПАКЕТОВ ПРИКЛАДНЫХ ПРОГРАММ ДЛЯ НЕОДНОРОДНЫХ МНОГОЯДЕРНЫХ ПРОЦЕССОРОВ

Специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат диссертации на соискание ученой степени кандидата технических наук

Санкт-Петербург – 2012

Работа выполнена в федеральном государственном бюджетном образовательном учреждении высшего профессионального образования «Санкт-Петербургский государственный политехнический университет» (ФГБОУ ВПО «СПбГПУ»)

Научный руководитель: доктор технических наук, профессор **Шейнин Юрий Евгеньевич**

Официальные оппоненты: **Устинов Сергей Михайлович** доктор технических наук, профессор, профессор кафедры информационных и управляющих систем ФГБОУ ВПО «СПбГПУ»

Помозова Татьяна Геннадьевна кандидат технических наук, доцент, начальник лаборатории ОАО «ЦНПО «Ленинец»

Ведущая организация: Федеральное государственное бюджетное учреждение науки Санкт-Петербургский институт информатики и автоматизации Российской академии наук

Защита состоится "20" декабря 2012 года в 16 часов на заседании диссертационного совета Д 212.229.18 ФГБОУ ВПО «Санкт-Петербургский государственный политехнический университет» по адресу: 195251, Санкт-Петербург, ул. Политехническая, д. 21, 9-й учебный корпус, ауд. 325.

С диссертацией можно ознакомиться в фундаментальной библиотеке ФГБОУ ВПО «Санкт-Петербургский государственный политехнический университет».

Автореферат разослан " 15 " ноября 2012 г.

Ученый секретарь
диссертационного совета Д 212.229.18
к.т.н., доцент **Васильев Алексей Евгеньевич**

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы. Развитие микропроцессорной техники в современных условиях тесно связано с использованием концепции многоядерности. На сегодняшний день сформировалось два направления: создание однородных многоядерных процессоров, в которых все ядра архитектурно идентичны и функционально симметричны, а также создание неоднородных многоядерных процессоров (НМП), в которых кристалл включает ядра, имеющие различную архитектуру и предназначенные для решения своего круга задач. Среди представителей НМП можно выделить процессоры для встраиваемых применений, построенные как «системы-на-кристалле». Вычислительные ядра таких процессоров имеют непосредственный доступ только к своей локальной памяти, объем которой составляет от нескольких десятков до нескольких сотен килобайт. В состав ВС входит модуль основной оперативной памяти объемом до одного гигабайта, в котором размещаются код и данные программы. Основным отличием указанных СнК является наличие независимо работающих ядер, предназначенных для организации информационного обмена (ЯОИО) между различными модулями памяти ВС. Основной проблемой в таких системах является организация параллельной обработки больших массивов данных в рамках локальной памяти вычислительных ядер с необходимостью явной организации информационного обмена.

Сложность и многообразие конфигураций многоядерных СнК постоянно возрастают. Создание высокопроизводительного пакета прикладных программ (ППП) для каждой из платформ требует значительных затрат ресурсов. Для снижения затрат на разработку и сопровождение ППП постепенно сформировался подход к их построению, в основе которого лежит автоматическая подстройка программы под имеющиеся ресурсы ВС. Вызов программы приводит к генерации системы заданий, связанных зависимостями по данным, а динамическое планирование их параллельного выполнения исключает необходимость модификации программы при переносе пакета на процессоры с другим числом ядер.

В изучение вопросов, связанных с эффективной организацией высокопроизводительных ППП и разработкой средств поддержки создания

параллельных программ, значительный вклад внесли такие ученые как Джек Донгарра, Фрэд Густавсон, Чарльз Лэйзерсон и др. Значительный вклад в развитие отечественных многоядерных структур, а также проектирование высокопроизводительных программных комплексов для них, внесли Я.Я. Петричкович, Т.В. Солохина, Ю.Н. Александров, А.А. Беляев, В.Д. Глушков, В.Ф. Никольский и др.

Существующие подходы направлены на построение ППП для однородных многоядерных СнК, и оказываются неприменимыми к большинству рассмотренных в работе СнК для встраиваемых применений. Анализ сложившейся ситуации говорит об актуальности разработки новых методов построения ППП.

На основании анализа общих черт существующих методов, а также, учитывая схожесть архитектуры многоядерных СнК, в работе предложен новый метод построения ППП. Отличие предложенного метода от рассмотренных аналогов состоит в том, что генерация системы фрагментов задачи и распределение ресурсов производятся до выполнения программы. Схожесть с существующими методами заключается в использовании модели статических потоков данных и идентичном подходе к декомпозиции задач. Определение структуры программы и распределение фрагментов задачи по узлам СнК до выполнения программы влечет за собой достижение высокой производительности обработки данных для задач малой размерности, что особенно актуально для встраиваемых систем, ресурсы которых ограничены. При переносе ППП на новые архитектуры использование внешнего по отношению к инструментальным средствам текстового описания модели НМП позволяет заменить его, не изменяя инструментальных средств.

Цель диссертационной работы – разработка методов, обеспечивающих автоматическую адаптацию программы к характеристикам задачи и неоднородной многоядерной «системы-на-кристалле» (НМСнК), а также высокую степень переносимости, при построении высокопроизводительных ППП, предназначенных для решения задач обработки клеточных матриц, которые можно сформулировать в виде совокупности алгебраических операций над их клетками, в рамках НМСнК, построенной на базе подсистемы памяти с программируемым информационным обменом.

Задачи исследования.

1. проанализировать существующие неоднородные многоядерные «системы-на-кристалле», сформировать их общий технический облик;
2. исследовать существующие методы построения пакетов прикладных программ для многоядерных «систем-на-кристалле», выявить их преимущества и недостатки;
3. выбрать, либо предложить формальную модель системы заданий позволяющую гибко производить распределение вычислительной нагрузки и информационного обмена между входящими в состав НМСнК ядрами;
4. разработать метод и алгоритм синтеза формального представления индивидуальной задачи обработки двумерных матриц большой размерности;
5. разработать формальную модель НМСнК;
6. разработать метод и алгоритм распределения вычислительной нагрузки и информационных обменов между узлами НМСнК.

Методы исследования. При разработке метода построения высокопроизводительных ППП использован аппарат теории графов и гиперграфов, теории множеств и теории расписаний. При синтезе макро-потоковых графов используется формализм графов потоков данных (data-flow graphs), при распределении ресурсов используется эвристический метод листового планирования.

Теоретические положения, выносимые на защиту, и их научная новизна.

1. предложена новая модель обобщенной задачи обработки двумерных числовых матриц – «свернутый» граф, содержащая информацию, достаточную для синтеза макро-потокового графа;
2. предложена новая методика построения «свернутого» графа по формуле в блочных обозначениях;
3. предложена новая модель системы заданий - макро-потоковый граф, отражающая необходимость проведения обработки данных в рамках локальной памяти вычислительного ядра, имеющей малый объем, а

также наличие ЯОИО, способных вести обмен асинхронно по отношению к обработке данных вычислительными ядрами;

4. предложен новый метод и разработан алгоритм статического синтеза макро-потокового графа для индивидуальной задачи по «свернутому» графу, позволяющий избавиться от соответствующих накладных расходов во время выполнения программы;
5. предложена новая гиперграфовая модель неоднородного многоядерного процессора, содержащая информацию о каналах информационного обмена, работающих под управлением ЯОИО, а также о модулях основной памяти и локальной памяти вычислительных ядер;
6. предложен эвристический метод и разработан алгоритм статического распределения ресурсов, базирующийся на построении списка узлов макро-потокового графа, представляющих вычислительную нагрузку в задаче, отличающийся от известных предоставлением гарантии отсутствия переполнения локальной памяти при успешном построении расписания.

Достоверность результатов. Достоверность полученных результатов, обеспечивается использованием математического аппарата теории множеств, теории расписаний, теории графов и гиперграфов, а также – доказательством утверждений. Для апробации результатов работы был создан опытный образец ППП, проведены натурные эксперименты на существующих образцах процессорной техники, а также эксперименты с разработанной имитационной моделью.

Практическая значимость. Разработан расширяемый опытный образец ППП, позволяющий свести долю ручного программирования к разработке подпрограмм для вычислительных ядер и разработке слоя абстрагирования от аппаратных механизмов конкретной «системы-на-кристалле». Распределение ресурсов в ППП производится на основании формальной модели НМСнК. На вход инструментария подается текстовый файл с описанием модели, что позволяет расширять возможности ППП на новые системы без изменения инструментальных средств.

В результате экспериментального исследования характеристик нескольких ключевых подпрограмм библиотеки BLAS выявлено, что

вычислительно-емкие подпрограммы достигают высокого уровня производительности уже при малых значениях размерности задачи, масштабирование вычислений близко к линейному, а генерация подпрограмм для задач большой размерности и НМП с большим количеством вычислительных ядер производится за приемлемое время.

Внедрение и реализация результатов работы. Основные результаты диссертационной работы использованы в ОАО НПЦ "ЭЛВИС" и ФНПЦ ОАО «НПО «Марс». Реализация научных положений и результатов диссертационной работы подтверждена соответствующими документами о внедрении.

Апробация работы. Основные положения и результаты диссертации докладывались и обсуждались на конференциях: «IX Международная научная конференция, посвященная 45-летию Сиб. гос. аэрокосмич. ун-та имени акад. М.Ф.Решетнева» (2005 год), «IEEE Tenth International Symposium on Consumer Electronics» (2006 год), «Шестой международный научно-практический семинар и молодежная школа «Высокопроизводительные параллельные вычисления на кластерных системах»» (2006 год), «Всероссийский форум студентов, аспирантов и молодых ученых «Наука и инновации в технических университетах» (2007 год), «7th conference of open innovations framework program FRUCT» (2010 год), «11th conference of open innovations framework program FRUCT» (2012 год).

Публикации. По материалам диссертации опубликовано девять печатных работ, две из которых – в издании, включенном в перечень ВАК. Одна работа опубликована в профильном журнале.

Структура и объем работы. Диссертация состоит из введения, четырех глав и заключения, списка использованных источников и девяти приложений. Диссертация содержит 174 страницы машинописного текста, включая 33 рисунка и 16 таблиц, а также приложения объемом 86 страниц, включая 25 рисунков и 5 таблиц. В списке использованной литературы 127 наименований.

СОДЕРЖАНИЕ РАБОТЫ

Во введении обоснована актуальность темы, определена цель и сформулированы задачи работы, показана научная новизна и практическая ценность полученных результатов.

<u>В первой главе</u> данной работы представлен обзор существующих представителей семейства неоднородных многоядерных процессоров (НМП), а также – обзор сформировавшихся на данный момент подходов к программированию ВС, построенных на базе НМП.

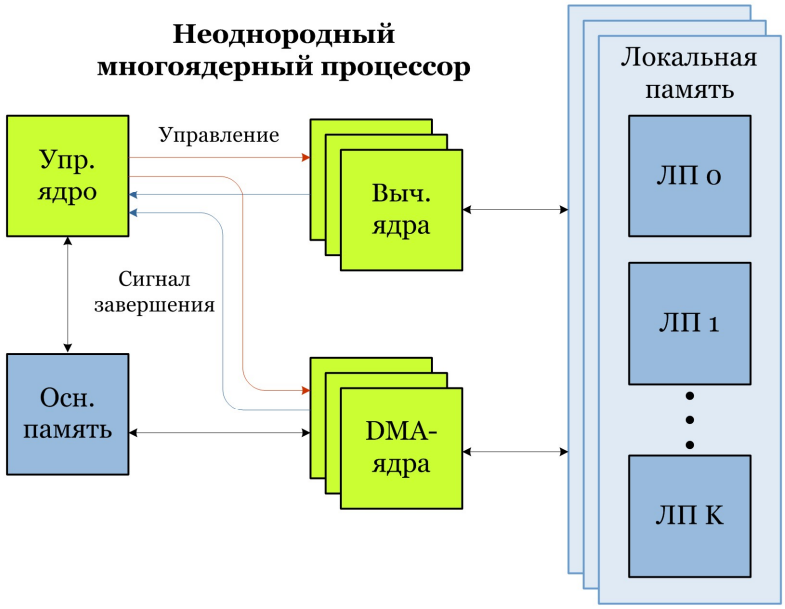

Рисунок 1. Обобщенная структурная схема НМП

Неоднородные многоядерные процессоры, рассмотренные в рамках данной работы, имеют общий технический облик (см. рис. 1) и обладают следующими отличительными характеристиками:

1. модуль основной ОП имеет размер от нескольких Мб до нескольких сотен Мб;
2. локальная память вычислительного ядра имеет объем от нескольких десятков до сотен Кб;
3. управляющее ядро, имеет доступ ко всем ключевым ресурсам системы;
4. каждое вычислительное ядро имеет доступ только к локальной памяти;
5. команды загрузки/сохранения данных и выборка команд в вычислительных ядрах работают в пределах поля локальной памяти;
6. для организации информационного обмена между различными модулями памяти требуется привлечение DMA-ядер (ЯОИО);
7. имеется возможность организации прямого обмена как между основной памятью и любым модулем локальной памяти, так и между любой парой модулей локальной памяти двух различных вычислительных ядер с привлечением хотя бы одного из DMA-ядер;

8. DMA-ядра способны работать асинхронно по отношению к работе других компонентов ВС.

Известные автору подходы к программированию НМП основаны либо на динамическом построении графа потоков данных, либо используют заданное «вручную» расписание. При использовании динамической схемы в процессе работы программа строит граф потоков данных, а динамический планировщик распараллеливает процесс обработки данных. Описанная подготовительная работа влечет за собой непроизводительные простои ресурсов НМП. При использовании статической схемы сервисная таблица, предназначенная для отслеживания прогресса обработки данных, должна располагаться в локальной памяти вычислительного ядра, что сильно ограничивает размерность решаемой задачи. Кроме этого, при использовании указанных подходов неявно предполагается, что у каждого вычислительного ядра есть собственное DMA-ядро и возможность управления им, что для некоторых из рассмотренных в данной работе НМП неверно.

Большое разнообразие архитектур НМП, а также присущая им сложность создания ППП и их переноса на вновь появляющиеся образцы, делает актуальным построение таких ППП, которые используют модель потоков данных для повышения гибкости при принятии решений в процессе распределения ресурсов ВС. Малая мощность встраиваемых систем делает целесообразной разработку легковесных алгоритмов синхронизации параллельных процессов, а также отказ от динамической генерации внутреннего представления параллельной программы и распределения ресурсов ВС по ходу выполнения программы в пользу полностью статической схемы.

Поскольку управление информационным обменом между модулями памяти ложится на DMA-ядра, целесообразно дополнить граф потоков данных узлами, представляющими информационный обмен для дальнейшего распределения заданий между DMA-ядрами. Для снижения непроизводительных потерь во время выполнения программы в данной работе предложена новая статическая схема распределения ресурсов, опирающаяся на разработанный метод статической генерации графа потоков данных для представления программы и предложенную формальную модель НМП.

Вторая глава содержит укрупненную схему предложенного в работе метода построения ППП и посвящена проблеме автоматического синтеза графового представления индивидуальной задачи обработки двумерных числовых матриц. Дано формальное определение «свернутого» графа и макро-потокового графа. Представлена методика построения «свернутого» графа и алгоритм синтеза макро-потокового графа, доказываются их свойства.

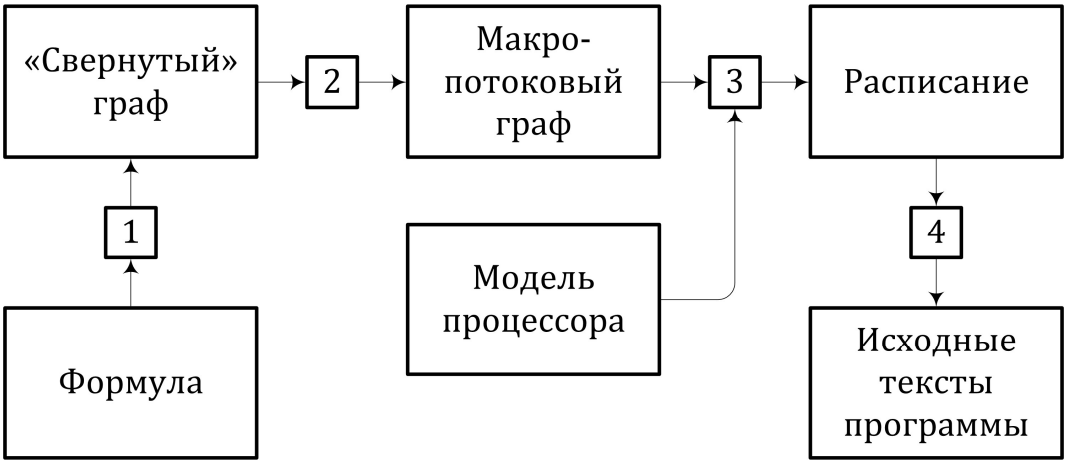

Рисунок 2. Укрупненная схема метода синтеза программы

На рис. 2 представлена укрупненная схема предложенного в данной работе метода синтеза исходных текстов параллельной программы. Разработанный метод применим для построения пакетов программ, предназначенных для решения задач обработки клеточных матриц, которые можно сформулировать в виде совокупности алгебраических операций над клетками матриц. Каждая индивидуальная задача может быть представлена макро-потоковым графом, являющимся крупногранулярным графом потоков данных. В основу предложенного формализма легли работы Денниса и Дэвиса.

Определение 1. Макро-потоковый граф – размеченный ориентированный ациклический граф потоков данных $G_A = <V_A, E_A>$ с дополнительным разбиением всего множества акторов V_A на множество акторов передачи данных V_D и множество вычислительных акторов V_C. Правила срабатывания акторов такие же, как в модели статических потоков данных. Разметка акторов позволяет адекватно описать процесс параллельной обработки блоков данных.

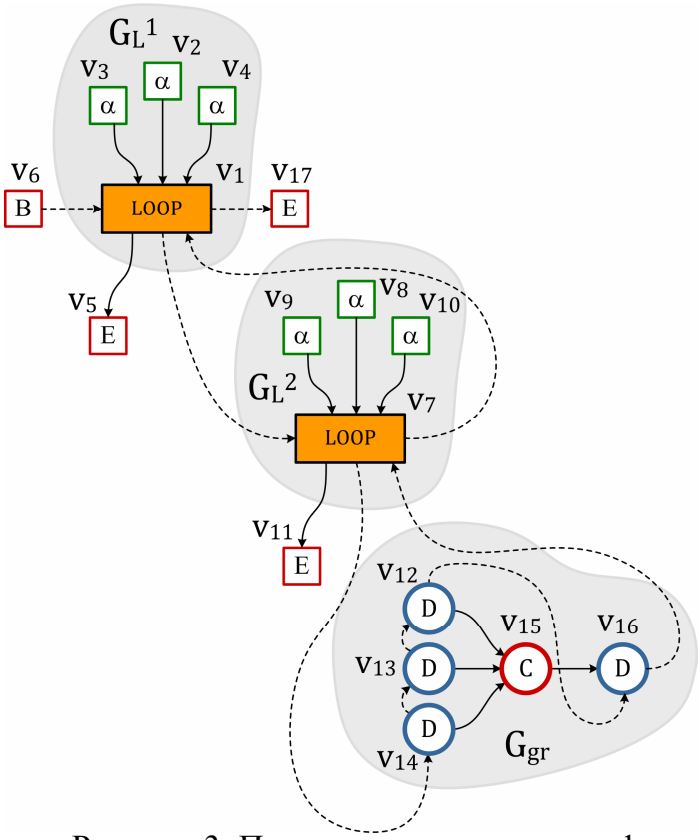

Рисунок 3. Пример «свернутого» графа

При этом, под актором передачи данных понимается актор, соответствующий передаче блока данных в процессе выполнения программы; под вычислительным актором – актор, соответствующий вызову вычислительной гранулы; под вычислительной гранулой – подпрограмма, предназначенная для обработки данных вычислительным ядром над полем своей локальной памяти.

Необходимым условием применимости разработанного метода построения ППП является существование формулы в блочных обозначениях, на вид которой наложены дополнительные ограничения, указанные в работе. По формуле F с использованием методики CONSTRUCT_FOLDED_GRAPH строится графовое представление обобщенной задачи обработки матриц - «свернутый» граф.

Определение 2: «Свернутый» граф – размеченный ориентированный граф $G_B = <V_B, E_B>$. Все множество вершин графа G_B можно разделить на взаимно-непересекающиеся подмножества $V_B = V_D \cup V_C \cup V_\alpha \cup B \cup E \cup L$, где: V_D – акторы передачи данных, V_C – вычислительные акторы, V_α – константные акторы, B – акторы, генерирующие управляющие токены, E – акторы, поглощающие токены, L – вершины-циклы. Множество дуг можно разделить на взаимно-непересекающиеся подмножества $E_B = E_D \cup E_C$, где: E_D – дуги передачи данных, E_C – дуги передачи управления.

Ключевым элементом, на базе которого строится любой «свернутый» граф, является вершина-цикл (v_1, v_7 на рис. 3). Назначение вершины-цикла заключается в представлении процесса перебора значений одной координатной переменной в аналитической записи решения задачи. Каждому квантору общности, а также каждому знаку сокращенной суммы (Σ) в формуле ставится

в соответствие вершина-цикл. Каждому равенству ставится в соответствие подграф (G_{gr} на рис. 3), представляющий подкачку исходных данных, вызов вычислительной гранулы и отвод результата. Остальные элементы добавляются в граф для обеспечения его дальнейшего преобразования в макро-потоковый граф.

Представленный на рис. 3 «свернутый» граф соответствует следующей формуле:

$$y_i^1 = \sum_{\forall j \in J} A_{ij}\, x_j + y_i \;\; \forall i \in I$$

, где $A \in R^{M \times N}$, $x \in R^N$, $y \in R^M$, I и J – множества блочных координат.

В результирующем «свернутом» графе (см. рис. 3) квантору общности соответствует подграф $G_L{}^1$, знаку сокращенной суммы – подграф $G_L{}^2$, равенству – подграф G_{gr}. Константные акторы (v_2-v_4, v_8-v_{10}) задают границы и шаг изменения координатных переменных i и j.

CONSTRUCT_FOLDED_GRAPH(F)

1 Добавить актор $v_b \in B$ и запомнить его
2 **while** в формуле F есть элементы **do**
3 **if** встретилась область действия ∀ **then**
4 Добавить подграф G_L, отражающий перебор значений координатной переменной и запомнить v_l, соответствующую ∀
5 Добавить связи с другими вершинами-циклами (v_l)
6 **else if** встретилось = без знака Σ **then**
7 Добавить G_{gr} и запомнить его
8 **else if** встретилось = со знаком Σ **then**
9 Добавить подграф G_L, отражающий перебор значений переменной суммирования и запомнить v_l, соответствующую Σ
10 Добавить связи с другими v_l
11 Добавить G_{gr}
12 Связать G_L и G_{gr} дугами
13 **end if**
14 Связать дугой последнюю пару запомненных элементов, если они не были связаны ранее
15 **if** закончилась область действия ∀ **then**

16 Связать дугой последний запомненный элемент и v_l, соответствующую $\forall$, и запомнить v_l

17 **end if**

18 **end while**

19 Добавить актор $v_e \in E$

20 Связать дугой последний запомненный элемент и v_e

Методика CONSTRUCT_FOLDED_GRAPH обладает важным свойством, указанным ниже.

Лемма 1. Для любой заданной формулы F, удовлетворяющей ограничивающим условиям, используя методику CONSTRUCT_FOLDED_GRAPH, можно построить конечный «свернутый» граф G_B.

Доказательство указанного свойства приведено в работе.

Каждой индивидуальной задаче ставится в соответствие кортеж параметров Ω, содержащий размерности задачи по каждому измерению, гранулярность разбиения матриц по каждому измерению и др. Указанный кортеж совместно со «свернутым» графом однозначно задает макро-потоковый граф.

Для синтеза макро-потокового графа используется представленный далее алгоритм.

MACROFLOW_GRAPH_SYNTH(G_B, Ω)

1 **while** в графе есть вершина-цикл v_l **do**

2 Произвести «развертку» вершины-цикла v_l

3 **if** вершина-цикл v_l соответствует Σ **then**

4 Произвести «склейку» соответствующих акторов передачи данных

5 **end if**

6 **end while**

7 Произвести «склейку» оставшихся акторов передачи данных

8 Удалить из графа все вспомогательные элементы

При синтезе макро-потокового графа производится последовательная «развертка» всех вершин-циклов. В результирующем графе остается по одному подграфу, представляющему подготовку данных и решение одного фрагмента задачи для каждого набора значений координатных переменных. Также, в процессе трансформации производится «склейка» акторов передачи данных,

т.е. добавление дуг между акторами, представляющими передачу промежуточных результатов через основную память.

Алгоритм MACROFLOW_GRAPH_SYNTH обладает указанными далее свойствами.

Теорема 1. При использовании алгоритма MACROFLOW_GRAPH_SYNTH по заданному кортежу параметров индивидуальной задачи Ω и соответствующему ему «свернутому» графу G_B, построенному по методике CONSTRUCT_FOLDED_GRAPH всегда можно построить конечный макро-потоковый граф G_A.

Теорема 2. Макро-потоковый граф G_A, построенный с использованием алгоритма MACROFLOW_GRAPH_SYNTH, является адекватным представлением индивидуальной задачи, которой соответствует формула F, при условии, что ранее по F был построен «свернутый» граф G_B и задан соответствующий ему кортеж параметров индивидуальной задачи Ω.

Доказательство указанных свойств приведено в работе.

Третья глава посвящена задаче распределения ресурсов НМП. Вводятся понятия гиперграфовой модели процессора и расписания. Представлено описание предложенного многоэтапного метода распределения ресурсов и детально рассмотрен каждый из этапов. Приведены и доказаны основные свойства предложенного метода.

Модель ВС, используемая в работах известных авторов, предполагает наличие у каждого процессора неограниченного количества доступной оперативной памяти. Для рассматриваемых в данной работе архитектур указанное допущение является некорректным, это и обуславливает необходимость разработки нового метода распределения ресурсов.

По синтезированному ранее макро-потоковому графу, с использованием гиперграфовой модели НМП $H_p = <V_p, E_p>$ производится распределение его ресурсов и строится расписание. В модели процессора управляющие ядра $S_p \subset V_p$, вычислительные ядра $C_p \subset V_p$ и DMA-ядра $D_p \subset V_p$, а также модули памяти $M_p \subset V_p$ представлены вершинами. Каналы информационного обмена $E_p^{IED} \subset E_p$, работающие под управлением специализированных ядер, представлены ориентированными гипердугами $e_i = <m_1, d_1, m_2>$, каждая из которых инцидентна двум вершинам $m_1, m_2 \in M_p$, представляющим модули

памяти, а также вершине $d_1 \in D_p$, представляющей ядро, ответственное за пересылку. Распределение функций управления ядрами, а также указание модулей памяти, способных хранить задания для ядер, ответственных за пересылки, также представлено в виде гипердуг. Полученное в ходе распределения ресурсов расписание преобразуется в исходные тексты программы.

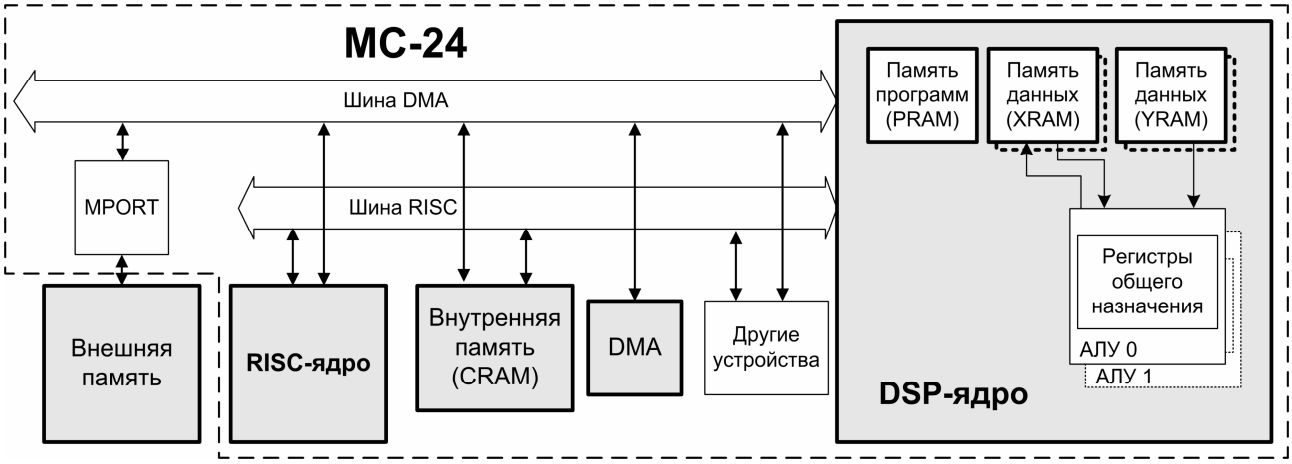

Рисунок 4. Структурная схема процессора MC24

Для процессора MC24, структурная схема которого представлена на рис. 4, структура множества V_p следующая: $S_p = \{s_1\}$, $C_p = \{c_1\}$, $D_p = \{d_1\}$, $M_p = \{m_1, m_2, m_3, m_4, m_5\}$, $E_p^{IED} = \{<m_2, d_1, m_3>, < m_2, d_1, m_4>, < m_2, d_1, m_5>, < m_4, d_1, m_2>\}$. При этом, соответствие между вершинами из M_p и модулями памяти следующее: m_1 – CRAM, m_2 – внешняя память, m_3 – PRAM, m_4 – XRAM, m_5 – YRAM. Поскольку память CRAM не участвует в обработке данных, каналы информационного обмена с ее участием в модели H_p не отражены.

Под расписанием в данной работе понимается пространственно-временное распределение акторов макро-потокового графа. Поскольку процесс обработки данных носит многоэтапный характер, и этапы отделяются друг от друга барьерной синхронизацией, вместо абсолютного времени используется номер этапа, в течение которого срабатывает каждый актор графа. При этом, для акторов передачи данных дополнительно определяется номер задания в цепочке соответствующего ядра, ответственного за пересылку.

RESOURCE_ALLOC(G_A, H_p)

1 Оценить достаточное количество памяти

2 Выделить память под буферы

3 Распределить вычислительную нагрузку

4 Произвести реструктуризацию макро-потокового графа

5 Организовать подкачку кода вычислительных гранул

6 Распределить информационные обмены

7 Распределить пространство буферов

При распределении ресурсов по макро-потоковому графу определяется достаточное количество локальной памяти вычислительных ядер, под каждый тип информации выделяется свой набор буферов, причем для перекрытия обработки и обменов применяется множественная буферизация памяти.

В процессе распараллеливания вычислений используется модификация алгоритма листового планирования. Вычислительные акторы выстраиваются в порядке обратной топологической сортировки. Акторы распределяются по вычислительным ядрам таким образом, что на каждом шаге алгоритма выбирается вычислительное ядро с наиболее «поздним» свободным этапом (ALAP). «Близость» вычислительных акторов в списке определяется затратами на передачу промежуточных результатов между соответствующими вызовами подпрограмм.

При распределении вычислительной нагрузки производится реструктуризация макро-потокового графа. Реструктуризация макро-потокового графа заключается в замене двух связанных акторов передачи данных на один, в том случае, когда передача блока данных через временный буфер в основной памяти заменяется на прямую межмодульную передачу данных, либо – передачу данных через локальную память вычислительного ядра. При реструктуризации графа должно соблюдаться указанное ниже условие.

Условие 1. Вычислительный актор-приемник данных срабатывает не позже, чем через n этапов после срабатывания актора-поставщика данных, где n – количество буферов данных.

При организации подкачки кода каждая загрузка гранулы назначается на этап, предшествующий ее использованию, привлекается первое подходящее ядро, организующее пересылку, подкачка осуществляется после завершения всех обменов данными в рамках соответствующего этапа.

Использование описанного выше правила не оказывает сильного влияния на длительность расписания при справедливости следующих

допущений: вычислительные гранулы целиком умещаются в локальной памяти; размеры гранул позволяют при решении задачи производить однократную и одномоментную загрузку гранул в локальную память вычислительных ядер по мере необходимости; в силу небольшого размера каждой гранулы, их подкачка оказывает малое влияние на длительность расписания.

Для распределения обменов данными сначала все обмены распределяются по этапам обработки данных, затем – внутри этапов, учитывая присущий транспортной подсистеме параллелизм. Далее представлен алгоритм межэтапного распределения обменов данными.

TRANSFER_TO_STAGE_MAP(G_A, H_p)

1 Выделить из V_D множества Q и L // Составить одноименные списки
2 **for all** v_i из L **do**
3 Произвести оценку значений ς_{min}, ς_{max}
4 **end for**
5 Произвести сортировку списка L в порядке невозрастания значений ς_{min}
6 Упорядочить этапы согласно максимуму вычислительной нагрузки в невозрастающем порядке следования значений
7 **for all** этапов **do** // С номером i при просмотре слева-направо
8 Построить список S акторов из L, которые могут сработать в течение i-го этапа
9 Произвести сортировку S в порядке невозрастания времени передачи данных
10 Распределить акторы из S по каналам информационного обмена, постепенно загружая каждый канал на время вычислений
11 **end for**
12 Распределить оставшиеся акторы из L по каналам информационного обмена, каждый раз выбирая канал с минимальной загрузкой
13 Распределить акторы из Q максимально «приближая» обмен данными к их обработке

В алгоритме TRANSFER_TO_STAGE_MAP используются следующие обозначения: множество $L = V_D \setminus Q \cup M$, $Q = Q_{in} \cup Q_{out}$, Q_{in} содержит акторы, соответствующие начальной загрузке данных в локальную память с

последующим началом вычислений в ядрах; Q_{out} содержит акторы, соответствующие окончательной выгрузке данных из локальной памяти с последующим завершением вычислений в ядрах; M содержит акторы передачи данных через локальную память. Под каналом информационного обмена понимается канал, связывающий одну пару модулей памяти (наличие нескольких управляющих ЯОИО игнорируется).

Каждому актору передачи данных ставится в соответствие интервал (в этапах) ρ, в течение которого актор может сработать.

$$\varsigma_{min}, \varsigma_{max}: V_D \rightarrow Z_0^+,$$

где ς_{min} – точная «левая» граница интервала ρ, ς_{max} – точная «правая» граница ρ. Правила оценки границ интервала ρ приведены в работе.

PER_STAGE_TRANSFER_SEQUENCING(G_A, H_p)

1 Составить список D
2 Упорядочить D по невозрастанию номеров этапов
3 **for all** этапов **do** // С номером i при просмотре слева-направо
4 Составить список R
5 Составить список Q
6 Упорядочить Q по невозрастанию суммарного времени обмена в каждой группе
7 **while** в Q есть группы **do**
8 Найти первую группу g_j, для которой количество конфликтующих с ней групп из L минимально
9 Назначить акторы из g_j на DMA-ядро, обеспечивающее минимум конфликтов и при этом - минимальное время запуска
10 Увеличить время освобождения выбранного DMA-ядра
11 Назначить акторам из g_j номера заданий в очереди
12 Увеличить номер нового задания выбранного DMA-ядра
13 Добавить g_j в L
14 Удалить g_j из Q
15 **end while**
16 **end for**

В алгоритме PER_STAGE_TRANSFER_SEQUENCING приняты следующие обозначения: D - список всех акторов передачи данных, соответствующих

обменам, требующим привлечения DMA-ядра; R - список, содержащий все акторы передачи данных, срабатывающие в течение заданного этапа; Q - список, содержащий все группы (группировка по каналам информационного обмена) акторов передачи данных, требующие распределения по DMA-ядрам в рамках заданного этапа; L - список, содержащий K групп акторов передачи данных, распределенных ранее, где K равно количеству DMA-ядер в процессоре.

Для описанного алгоритма распределения ресурсов НМП справедлива указанная ниже теорема.

Теорема 3. Расписание, построенное с использованием алгоритма RESOURCE_ALLOC, является допустимым.

Доказательство теоремы приведено в работе.

В четвертой главе работы представлен анализ данных экспериментов с четырьмя подпрограммами, входящими в состав библиотеки BLAS.

Для вычислительно-емких подпрограмм (strsm, sgemm) были получены следующие экспериментальные результаты:

- ускорение при распараллеливании вычислений на два ядра процессора MC0226 составляет от 1,42 для strsm с размерностью 152 элемента, до 1,97 для sgemm с размерностью 168 элементов;
- производительность подпрограммы sgemm при использовании двух вычислительных ядер процессора MC0226 составляет: 233 МФлоп/с (72,8% от пиковой производительности двух ядер, что составляет 85% от максимальной производительности двух гранул) для размерности задачи равной 84 элементам и 264 МФлоп/с (82,5% от пиковой производительности двух ядер, что составляет 96,4% от максимальной производительности двух гранул) для размерности задачи равной 168 элементов. Разрыв между производительностью гранул и производительностью программы обусловлен наличием синхронизации, подкачки данных в начале и отвода результатов в конце процесса их обработки и не может быть сокращен.

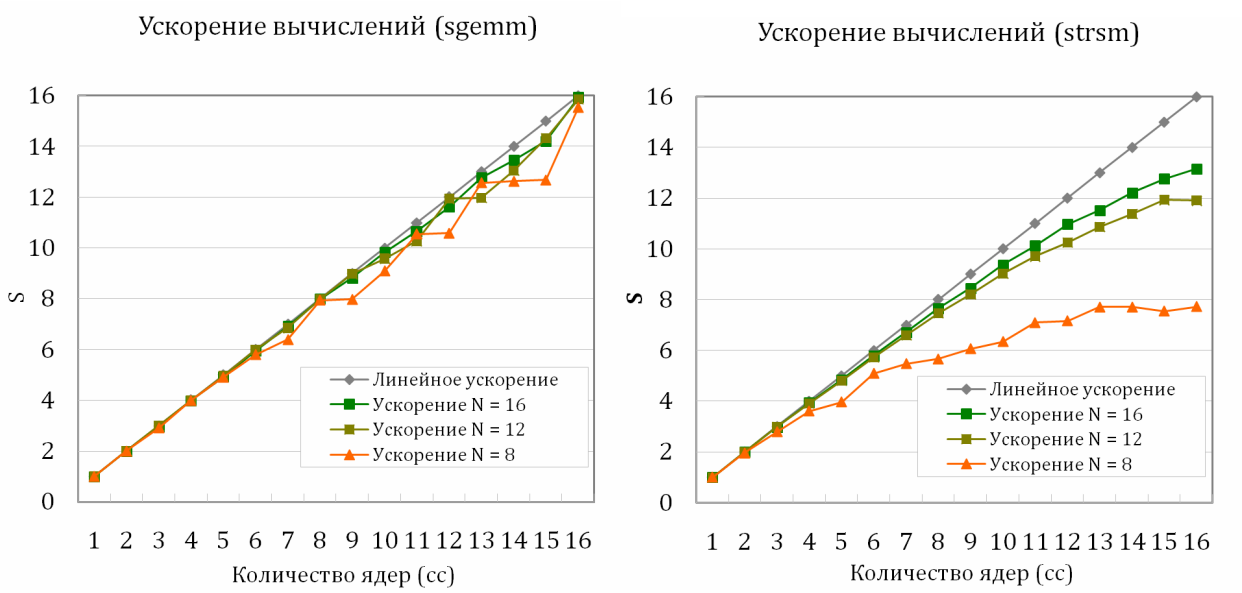

Рисунок 5. Графики масштабирования вычислений

В результате имитационного моделирования поведения подпрограмм sgemm и strsm на синтетической модели процессора, содержащей 16 вычислительных ядер, получены следующие результаты:

- масштабируемость указанных подпрограмм близка к линейной (см. рис. 5, где N – блочная размерность задачи);
- производительность подпрограммы strsm достигает 91% от производительности вычислительной гранулы sgemm.

В процессе разработки прототипа ППП были получены следующие результаты:

- перенос прототипа ППП между процессорами МС24 и МС0226 потребовал изменения менее чем 0,5% от общего объема инструментальных средств пакета;
- добавление в прототип каждой из четырех подпрограмм потребовало добавления менее чем 1,5% от общего объема инструментальных средств пакета для каждой подпрограммы.

В заключении приведены основные результаты проведенного исследования и их соотношение с целью и задачами, научной новизной, практической ценностью и положениями, выносимыми на защиту, поставленными и сформулированными во введении.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

Основные результаты работы можно сформулировать следующим образом:

1. на основе анализа архитектуры основных представителей семейства неоднородных многоядерных процессоров сформирован их общий технический облик, а также определен круг задач, являющихся специфичными для таких архитектур;
2. проведен обзор существующих подходов к построению пакетов прикладных программ для многоядерных архитектур. Выявлено, что использование модели потоков данных является перспективным, поскольку позволяет гибко подстраивать вычислительный процесс под возможности ВС. Обоснована необходимость разработки нового метода построения пакетов прикладных программ;
3. предложена формальная модель обобщенной задачи обработки двумерных числовых матриц;
4. предложена методика построения экземпляров модели обобщенной задачи обработки двумерных числовых матриц;
5. предложена формальная модель системы заданий, являющаяся расширением широко известной модели потоков данных и раскрывающая необходимость явной организации информационного обмена между модулями памяти;
6. разработан алгоритм синтеза формального представления индивидуальной задачи обработки двумерных числовых матриц большой размерности, доказана корректность разработанного алгоритма;
7. предложена гиперграфовая модель неоднородного многоядерного процессора, отражающая устройство транспортной подсистемы и подсистемы памяти;
8. разработан эвристический алгоритм распределения ресурсов, в основу которого положен листовой алгоритм распределения вычислительной нагрузки. Предложенный алгоритм предоставляет гарантии отсутствия переполнения памяти. При распределении памяти используется механизм множественной буферизации.

Таким образом, решены все задачи, поставленные для достижения сформулированной в работе цели.

В результате экспериментального исследования характеристик наиболее представительных подпрограмм из библиотеки BLAS выявлено, что вычислительно-емкие подпрограммы достигают высокого уровня производительности уже при малых значениях размерности задачи, масштабирование вычислений близко к линейному, а генерация подпрограмм для задач большой размерности и НМП с большим количеством вычислительных ядер производится за приемлемое время. Портирование пакета на новую платформу и добавление новых подпрограмм в пакет не требует привлечения существенных трудозатрат. Таким образом, цель работы достигнута.

На основе решенных в диссертации задач можно определить следующие **направления дальнейших исследований**:

1. расширение класса задач обработки матриц за счет совершенствования метода синтеза формального представления индивидуальной задачи;
2. отказ от использования барьерной синхронизации процессов;
3. совершенствование метода распределения памяти с целью добавления возможности повторного использования предварительно загруженных исходных данных;
4. учет возможности межъядерного обмена результатами вычислений через локальную память для тех процессоров, в которых вычислительные ядра имеют общее поле локальной памяти;
5. оценка отклонения длины расписания от длины оптимального расписания;
6. совершенствование метода распределения ресурсов неоднородного многоядерного процессора.

ПУБЛИКАЦИИ ПО ОСНОВНЫМ РЕЗУЛЬТАТАМ ДИССЕРТАЦИИ

1. **Недоводеев К.В. Метод генерации графов потоков данных, используемых при автоматическом синтезе параллельных программ для неоднородных многоядерных процессоров // Научно-технические ведомости СПбГПУ. – 2012. – №3. – Т. 2. – С. 47-52.**

2. Nedovodeev K.V. Adaptive libraries for multicore architectures with explicitly-managed memory hierarchies // 11th conference of open innovations framework program FRUCT: conference proceedings. – Saint-Petersburg: SUAI, 2012. – P. 126-135.

3. Nedovodeev K.V. Self-adapting software as a means of meeting the multicore challenge // 7th conference of open innovations framework program FRUCT: conference proceedings. – Saint-Petersburg: SUAI, 2010. – P. 83-86.

4. **Недоводеев К.В. Метод синтеза блочного алгоритма по его графовому представлению // Научно-технические ведомости СПбГПУ. – 2007. – №4. – Т. 2. – С. 141-148.**

5. Недоводеев К.В. Применение модели потоков данных для описания (макро)блочных алгоритмов // Всероссийский форум студентов, аспирантов и молодых ученых «Наука и инновации в технических университетах»: материалы всероссийского форума. – СПб: Изд-во СПбГПУ, 2007. – С. 53-54.

6. Недоводеев К.В. Организация (макро)потоковых вычислений в неоднородных мультиядерных процессорах // Высокопроизводительные параллельные вычисления на кластерных системах: сб. науч. тр. / под ред. проф. Р.Г. Стронгина – СПб.: Изд-во СПбГУ, 2007. – Т. 2. – С. 72-77.

7. Недоводеев К.В., Шейнин Ю.Е. Высокопроизводительная обработка больших массивов данных в неоднородных мультиядерных процессорах // Электронные компоненты. – 2006. – №9. – С. 116-122.

8. Nedovodeev KV. Multimedia Data Processing On Dual-Core Soc Multicore-24 // IEEE Tenth International Symposium on Consumer Electronics (ISCE 2006): conference proceedings. – NY: IEEE, 2006. – P. 142-147.

9. Недоводеев К.В. Организация взаимодействия RISC- и DSP-ядер сверхбольшой интегральной схемы «Мультикор» в задачах обработки сигналов // IX Междунар. науч. конф. посвящ. 45-летию Сиб. гос. аэрокосмич. ун-та имени акад. М.Ф. Решетнева: материалы междунар. науч. конф.. – Красноярск: Изд-во СибГАУ, 2005. – С. 293-294.