Организация конвейерно-параллельных вычислений для обработки потоков данных[1]

А.А. Талалаев

Аннотация. В работе рассматриваются принципы организации конвейерно-параллельной обработки потоков целевой информации, использованные при разработке нейросетевой программной системы космического назначения. Предлагаемые решения позволят обеспечить эффективное выполнение задач на мультипроцессорных вычислительных системах.

Ключевые слова: программная система, ковейерно-параллельные вычисления, кластер, диспетчеризация.

Введение

Программная система (ПС) «Нейросеть» предназначена для обработки потоков данных телеметрии и снимков ДЗЗ, поступающих на наземные станции командно-измерительных систем (НС КИС). Общие принципы построения ПС «Нейросеть» являются дальнейшим развитием идей, заложенных в систему ПС «ИНС» [1]. Практика использования первой версии системы, показала недостаточно высокую масштабируемость вычислений при обработке больших объемов данных из-за отсутствия механизма планирования загрузки вычислительных узлов. Целью настоящей работы является разработка эффективного подхода для обработки данных на основе планирования загрузки узлов многопроцессорных систем. В данной работе предлагается система диспетчеризации, обеспечивающая конвейерно-параллельный режим обработки потоков данных на кластерных вычислительных устройствах (КВУ).

1. Архитектура ПС «Нейросеть» и принципы описания задач обработки данных

Архитектура ПС «Нейросеть» [2] является модульной, что позволяет адаптировать ее для использования на мультипроцессорных системах произвольного типа. Архитектура системы отвечает требованиям гибкости и расширяемости. Выбранная схема вычислительного ядра обеспечивает связь модулей, реализующих различные алгоритмы, предназначенные для решения прикладной задачи. По сути, ПС «Нейросеть» представляет собой комплекс инструментальных средств, поддерживающий создание программных систем, относящихся к различным предметным областям. В случае, если система не обеспечивает достаточной функциональности для решения конкретной прикладной задачи, программист может реализовать новые модули и включать их в требуемые схемы обработки данных.

[1]Работа выполнена в рамках Программы Союзного государства «Космос-НТ» (проект «Разработка прототипа программной нейросетевой системы контроля телеметрической информации, диагностики подсистем космических аппаратов, обработки космических снимков» и при поддержке РФФИ (проекты № 09-07-00006-а, № 08-01-00485-а).

Введем некоторые обозначения:

- «Модуль» - объект, реализующий некоторую функцию обработки данных. Значения аргументов реализованной функции определяются данными, поступающими на входы модуля по каналам передачи данных, выходные значения, если они определены, передаются на выходы модуля;
- «Канал передачи данных» - объект, связывающий модули, использованные для решения прикладной задачи, и обеспечивающий процесс пересылки данных между ними;
- «Схема обработки данных» – древовидная схема, которая позволяет средствами графического интерфейса задать произвольный набор модулей и связей между ними (каналов передачи данных), необходимых для решения прикладной задачи;
- «Вычислитель» - объект, копии которого существуют на каждом из доступных вычислительных узлов, принимает команды от планировщика и исполняет их, отсылая планировщику уведомления о выполнении. Имеет пул потоков, в которых, могут быть выполнены функции загруженных модулей;
- «Планировщик» - объект, существующий в единственном экземпляре, хранит информацию о доступности на узлах модулей обработки, состоянии вычислителей (свободен/занят) и данных, готовых для обработки.

На формализации понятия схемы обработки данных следует остановиться более подробно, т.к. оно является основополагающим. Схемы обеспечивают гибкость и расширяемость системы, позволяя адаптировать ее к различным прикладным областям. Управление процессом передачи данных между модулями входящими в схему – является основной задачей планировщика ПС.

В качестве примера, рассмотрим тестовую схему, содержащую шесть модулей M1,…, M6, использующих три атомарных ресурса (с именами A, B и C) и два «составных» ресурса (поименованных как AB и BC). Каждый из модулей в данной схеме имеет один вход (с именем ‘In’) или один выход (с именем ‘Out’), ресурс ‘AB’ имеет поля с типами ‘A’ и ‘B’, а ‘BC’ − поля типов ‘B’ и ‘C’.

В Табл. 1 приведено, в упрощенном виде, описание тестовой задачи в формате xml, принятом в ПС.

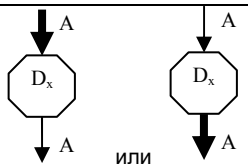
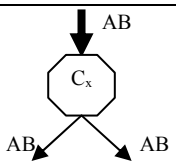
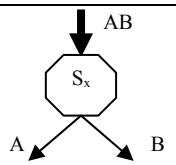
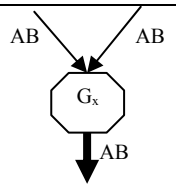
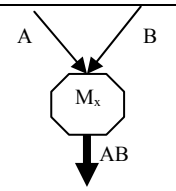
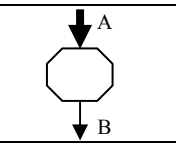
Схему обработки данных удобно представить в виде некоторого ориентированного графа, узлами которого являются модули, а дугами − заданные пользователем каналы передачи данных. Графическое представление схемы, описанной в Табл. 1, приведено на Рис. 1. Схема состоит из узлов ассоциированных с заданными модулями обработки и дуг, определяющих связи между ними.

Существует взаимнооднозначное соответствие между тестовым и графическим представлениями схемы задачи. Более того, все каналы, в общем случае, выполняют обработку в процессе передачи данных и в свою очередь описываются соответствующими графическими структурами. Всего определено шесть базовых операций над данными, передаваемыми между модулями, их описание приведено в Табл. 2.

Табл. 1. Описание связей между модулями (тестовая схема)

```
<?xml version="1.0"?>
<task>
        <modules>
                <module internalname="M1" name="m_1.so"/>
                <module internalname="M2" name="m_2.so"/>
                <module internalname="M3" name="m_3.so"/>
                <module internalname="M4" name="m_4.so"/>
                <module internalname="M5" name="m_5.so"/>
                <module internalname="M6" name="m_6.so"/>
        </modules>
        <channels>
                <channel>
                        <out>M1.Out</out>
                        <in>M2.In.A</in>
                </channel>
                <channel>
                        <out>M3.Out.B</out>
                        <in>M2.In.B</in>
                </channel>
                <channel>
                        <out>M3.Out.C</out>
                        <in>M4.In</in>
                </channel>
                <channel>
                        <out>M5.Out</out>
                                <in>M4.In</in>
                </channel>
                <channel>
                        <out>M5.Out</out>
                        <in>M6.In</in>
                </channel>
        </channels>
</task>
```

Табл.2. Операции над данными и их графическое представление

Наименование операции	Графическое представление	Краткое описание
Прямое соединение (Direct)		Операция буферизации данных без обработки.
Блок копирования ресурса (Copy)		Операция рассылки копий объекта на множество выходов.
Блок разделения ресурса (Split)		Операция выделения составных частей ресурса и их рассылки получателям.
Блок объединения данных (Gather)		Операция сборки в единый поток однотипных ресурсов поступающих по нескольким входам.
Блок сборки ресурса (Merge)		Операция сборки ресурса из множества частей поступивших на входы.
Блок преобразования типа ресурса		Базовая операция, определяющая возможность изменения типа представления данных.

Не останавливаясь на деталях алгоритма преобразования исходной схемы, приведем результирующую графическую структуру (Рис. 2), определяющую схему решения тестовой задачи.

На Рис.2 в виде прямоугольников показаны узлы, ассоциированные с модулями, использованными для решения задачи, а в виде шестиугольников – объекты, производящие преобразование данных и реализующие каналы передачи данных. Следует отметить, что, каналы выполняют дополнительную функцию, кэшируя промежуточные результаты обработки и сохраняя их в оперативной памяти или в низкоуровневой базе данных, что позволяет системе поддерживать сбалансированную загрузку вычислительных ресурсов в условиях ограничений на их число.

Схема (Рис. 2) описывает конвейеры обработки данных и содержит узлы следующих типов:

- «Поставщики» - модули, получающие данные извне и подающие их на обработку;
- «Обработчики» - в этой роли могут выступать заданные в схеме модули производящие обработку данных или преобразование данных из одного типа в другой (включая узлы, реализующие каналы передачи);
- «Потребители» - в качестве таких узлов выступают модули, не имеющие выходов, они завершают обработку данных (такие как сохранение результатов обработки или передача результатов обработки пользователю).

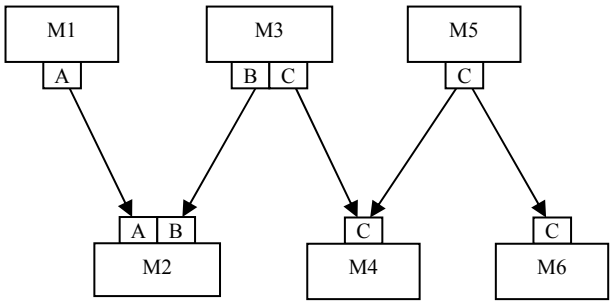


Рис. 1. Графическое представление тестовой схемы (исходное)

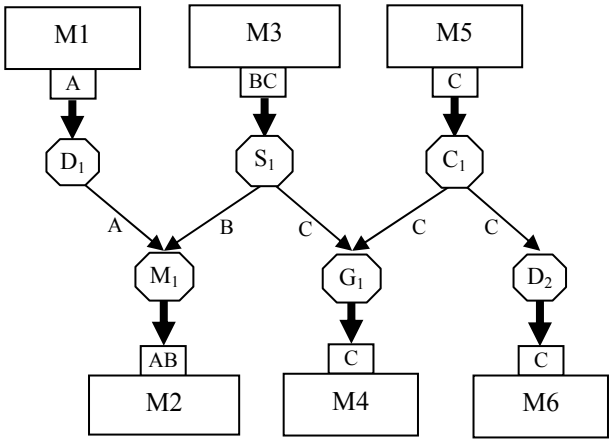


Рис. 2. Графическое представление тестовой схемы (после преобразования)

2. Принципы организации параллельно-конвейерных вычислений

Основной задачей планировщика является управление процессами передачи данных между модулями и инициализации их запуска путем передачи вычислителям управляющих команд. Планировщик решает задачу максимизации числа одновременно выполняемых команд в условиях ограниченных аппаратных ресурсов, используя «жадную» стратегию управления [3].

Общая схема взаимодействия Планировщик↔Вычислители↔Модули, представлена на Рис. 3. Введены следующие обозначения: S – планировщик, W_j – вычислитель, ассоциированный с узлом j, m_{ij} – модуль i, загруженный на узел j.

Схема определяет связи следующих типов:

1) «Планировщик↔Вычислители» – связь, определяющая передачу управляющих команд планировщика и подтверждений от вычислителей.

2) «Вычислитель↔Модули» и «Вычислитель↔Вычислитель» – связи, обеспечивающие параллельно-конвейерную обработку данных средствами ядра системы, реализуя межмодульное взаимодействие.

3) «Модуль↔Модуль» – связи, описывающие взаимодействие копий параллельных модулей, находящихся на разных узлах.

Поясним, как схема задачи выполняется на кластерной установке с применением планировщика, для чего рассмотрим другую, более простую, тестовую задачу. Пусть имеются три модуля m_1, m_2, m_3, причем: m_1– является поставщиком данных, m_2 – обработчиком, а m_3 – потребителем. Предположим также, что планировщику доступны три простейших вычислителя W_1, W_2, W_3. Это означает, что можно организовать несколько конвейеров обработки данных связанных друг с другом, как это показано на Рис. 4а. При этом данные на каждом из этапов обработки могут быть отправлены на любой из доступных вы-

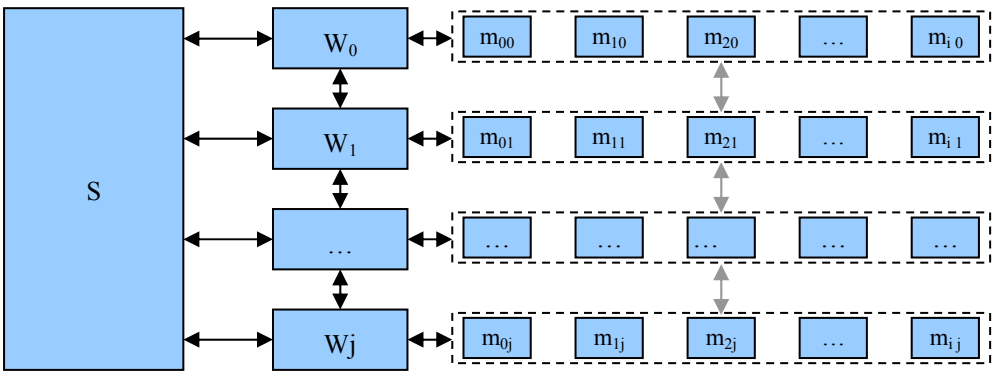


Рис. 3. Схема взаимодействия компонент системы диспетчеризации

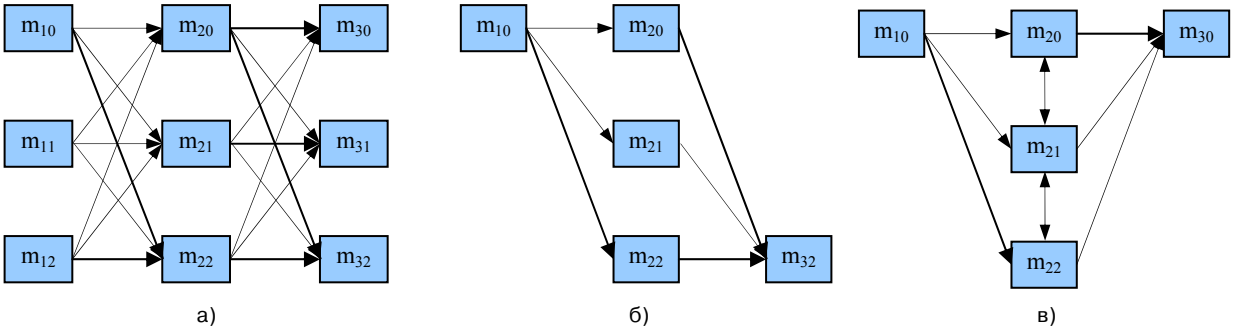


Рис. 4. Примеры схем решения задач управляемых планировщиком

а) идентичные конвейеры на узлах
б) части конвейеров на узлах
в) с использованием параллельного модуля

Табл. 3. Описание состояния системы

	W_0	W_1	W_2
m_1	0	0	0
m_2	0	0	0
m_3	0	0	0
Threads	1	1	1

а)

	W_0	W_1	W_2
m_1	0		
m_2	0	0	0
m_3			0
Threads	1	1	1

б)

	W_0	W_1	W_2
m_1	0		
m_2	0	0	0
m_3	0		
Threads	1	1	1

в)

числителей. Осуществляя привязку модулей, входящих в схему, к архитектуре КВУ можно проводить более тонкую настройку вычислительного процесса. Так, например, на различных узлах можно построить отдельные части конвейера (Рис. 4 б). Наличие в схеме задачи параллельных модулей (модулей, имеющих внутренний параллелизм, реализованный самим разработчиком) требует синхронизации времени их запуска. Если, предположить, что модуль m_2 является параллельным, то схема задачи примет вид, показанный на Рис. 4 в.

Планировщик может быть описан в виде конечного автомата, состояние которого определяется следующими составляющими:

1. Таблица, определяющая загруженность вычислителей. Данная таблица формируется по описанию задачи и описанию кластерной установки. Таблица хранит информацию о работающих, в настоящий момент, модулях и наличии неактивных потоков в пулах вычислителей. Количество потоков в пулах вычислителей на каждом из доступных узлов определяется пользователем и напрямую зависит от возможностей имеющегося аппаратного обеспечения.

Схемам, приведенным на Рис. 4 а,б,в соответствуют Табл. 3 а,б,в.

2. Информация о данных доступных для обработки. В ходе работы системы модули создают и передают на свои выходы пакеты данных которые ядро системы пересылает далее по схеме. Планировщик имеет информацию о всех пакетах хранящихся на вычислительных узлах, и использует ее для генерации новых управляющих команд.

3. Множество управляющих команд. На основе информации о доступных аппаратных ресурсах и имеющихся в настоящий момент готовых для обработки пакетах данных планировщик генерирует управляющие команды для вычислителей. Однако, в ходе работы системы, могут возникать ситуации, когда запуск модуля невозможен в текущий момент времени. Управляющие команды в этом случае буферизуются планировщиком, их выполнение откладывается до момента, когда необходимые ресурсы становятся доступными.

Все сообщения генерируемые ядром системы, как управляющие команды от планировщика к вычислителям, так и подтверждения от вычислителей о завершении обработки того или иного пакета данных, переводят систему в новое состояние.

Алгоритм работы планировщика итеративен и выглядит следующим образом:

```
Пока в схеме решения задачи существуют модули не завершившие работу
{
    Если в множестве команд существуют команды применимые в текущем состоянии
    {
            Выбрать применимую команду и передать ее вычислителю
    } иначе
    {
            Ждать подтверждения от вычислителя
    }
    Изменить состояние системы
}
```

Для модулей входящих в схему, устанавливается различный приоритет в обслуживании. Наивысшим приоритетом обладают параллельные модули, так как они требуют наибольшее количество аппаратных ресурсов. Приоритет непараллельных модулей определяется их позицией в схеме решения задачи (чем ближе модуль находится к концу конвейера, тем большим приоритетом он обладает). Данная стратегия способствует уменьшению задержки выдачи обработанных данных при работе в потоковом режиме.

Выводы

Построение оптимальной функции диспетчеризации в значительной степени осложнено тем, что, ядро системы не имеет априорных оценок о времени, необходимом модулям, входящим в схему, на обработку пакетов данных. При таких условиях предложенный подход выглядит достаточно эффективным решением, позволяющим обеспечить балансировку нагрузки вычислительных узлов КВУ. В качестве дополнительного механизма позволяющего повысить эффективность вычислений, предлагается использовать кэширование промежуточных данных.

Литература

1. Талалаев А.А. Особенности архитектуры параллельной программной системы распознавания графических образов на основе искусственных нейронных сетей – «Нейрокомпьютеры: разработка, применение», 2008, №9, с.43-51
2. Константинов К.А., Талалаев А.А., Тищенко И.П., Хачумов В.М. Концепция архитектуры нейросетевой системы контроля, диагностики и обработки изображений космического назначения – «Авиакосмическое приборостроение», 2009, №5, с.39-47
3. Коуги П. М. Архитектура конвейерных ЭВМ – М.: Радио и связь, 1985

Талалаев Александр Анатольевич. Младший научный сотрудник Института программных систем им. А.К. Айламазяна РАН. Окончил университет имени А.К. Айламазяна города Переславль-Залесский в 2006 году. Автор 17 печатных работ. Область научных интересов: искусственный интеллект, машинная графика, параллельные вычисления. E-mail: arts@arts.botik.ru.