05.13.11

А.А. Талалаев к.т.н., В.П. Фраленко к.т.н.

Федеральное государственное учреждение науки
Институт программных систем им. А.К. Айламазяна Российской академии наук,
Исследовательский центр мультипроцессорных систем,
Переславль-Залесский, alarmod@pereslavl.ru

КОМПЛЕКС ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ ПРОЕКТИРОВАНИЯ НЕЙРОСЕТЕВЫХ ПРИКЛАДНЫХ СИСТЕМ[1]

Работа посвящена разработке комплекса инструментальных средств для проектирования нейросетевых прикладных систем. Комплекс позволяет абстрагироваться от деталей низкоуровневой реализации параллельных алгоритмов организации нейросетевых вычислений. Программа в рамках комплекса, ориентированная на конвейерно-параллельную архитектуру, может использовать различные виды параллелизма одновременно. Предложенная архитектура позволяет полноценно использовать в том числе и ресурсы гетерогенной среды GPU-кластера. Приведены результаты экспериментальных исследований.

Ключевые слова: *комплекс проектирования, нейронная сеть, графический ускоритель, параллелизм.*

Введение

Современные требования к качеству и быстродействию анализа и обработки различных потоков данных (телеметрии, видеоинформации, звуковых и радиотехнических сигналов, потоков изображений и др.) обуславливают необходимость поиска единых подходов к построению интеллектуальных информационных технологий (в том числе и нейросетевых) и перспективных прикладных систем на их основе.

В настоящее время существует ряд разработок, представляющих собой высокоуровневые инструментальные комплексы проектирования прикладных систем. Особо стоит отметить такие технологии и системы как Kepler, ANSYS EKM, Microsoft Workflow Foundation, CODE и пр. [1-5]. Некоторые из них уже сейчас обладают возможностями по интеграции с системами кластерных вычислений, однако, ни одна из них не поддерживает концепцию конвейерно-параллельных вычислений в полном объеме. Разрабатываемый метод организации конвейерно-параллельных вычислений и экспериментальный образец программно-инструментального комплекса на его основе предназначен для поддержки высокопроизводительных вычислений на прикладных задачах пользователя. При этом большое внимание посвящено разработке и использованию средств проектирования прикладных нейросетевых систем.

Вопросами обработки данных на искусственных нейронных сетях (ИНС) ранее занимались Rosenblatt F., Kohonen T.K., Hopfield J.J., Verma B., Haykin S., Mahoney M., Cheng H., Wosserman F., Горбань А.Н., Галушкин А.И., Новосельцев В.Б., Ясницкий Л.Н. и другие исследователи. С помощью нейросетей можно прогнозировать временные ряды, выполнять распознавание оптических и звуковых сигналов, создавать самообучающиеся системы, управлять подвижными объектами и т.д. Для указанных задач характерны высокая размерность пространства признаков и критический фактор времени, что требует применения высокопроизводительной вычислительной техники. Разрабатываемые в рамках проводимого исследования технологии учитывают особенности ИНС, возможности

[1] Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект № 12-01-31500-мол_а «Разработка инструментальных программных средств для проектирования нейросетевых прикладных систем»).

современных технологий параллельного программирования и аппаратных средств отечественных кластерных вычислительных устройств (КВУ) семейства «СКИФ» [6].

Известно, что наиболее интенсивная вычислительная нагрузка приходится на фазу обучения нейронных сетей (например, сетей прямого распространения по алгоритму обратного распространения ошибки). Уменьшение времени обучения нейронной сети является актуальной задачей, особенно, на стадии экспериментального исследования пригодности тех или иных нейросетевых моделей, выбора оптимальных параметров для их успешного применения на практике. Наиболее ресурсоемкими операциями в нейровычислениях вообще и в алгоритме обратного распространения ошибок (error back propagation), в частности, являются матричные операции (скалярное умножение, умножение матрицы на вектор и т.п.). Для алгоритма Левенберга-Марквардта, используемого в обучении нейросетей, в том числе наиболее ресурсоемким оказывается и операция обращения матриц. Существуют хорошо отработанные методы распараллеливания таких вычислений на распределенных вычислительных системах. В частности, можно рекомендовать использование библиотеки OpenBlas [7], применение графических ускорителей (OpenCL, CUDA) [8,9].

Разрабатываемое алгоритмическое и программное обеспечение направлено на создание программно-инструментальной среды, позволяющей при разработке прикладных систем абстрагироваться от деталей низкоуровневой реализации параллельных алгоритмов декомпозиции нейросетевых вычислений и обеспечивающее достаточно высокие показатели алгоритмов конвейерно-параллельных вычислений по быстродействию при минимальных временных затратах на их разработку с использованием инструментальных средств. При этом учитывается возможность кроссплатформенной реализации с сохранением принципов универсализма, гибкости и расширяемости программного комплекса.

1. О разработанном программном комплексе

Разработанный программный комплекс (ПК) состоит из ядра комплекса, библиотеки программных модулей обработки целевой информации (в том числе модули нейронных сетей: прямого распространения, Хопфилда, Хемминга и Кохонена, неокогнитрон Фукушимы, вероятностная и сверточная нейронные сети), базы знаний, содержащей экспертную информацию о решаемой задаче, базы данных, содержащей обрабатываемую информацию, и интерфейсов пользователя и эксперта знаний. Архитектура комплекса основывается на предложенной ранее концепции [10].

Программный комплекс имеет прямую поддержку вычислений с использованием графических ускорителей (GPU) []. Программные библиотеки, использующие GPU (MAGMA, CULA, ViennaCL и др.), обеспечивают значительное преимущество при вычислениях за счет распараллеливания задач в случае, если возможна их декомпозиция на низком уровне. Одновременное использование набора GPU достаточно просто реализуется на системах с SMP-архитектурой, но может быть неэффективно на системах кластерного типа. Для использования GPU в рамках разработанного программного комплекса достаточным является выполнение единственного требования – библиотека алгоритмов GPU-вычислений должна обладать потокобезопасной реализацией (программный код является потокобезопасным, если он функционирует корректно при использовании нескольких параллельно запущенных вычислительных потоков). Указанное требование выполняется как в случае применения низкоуровневых библиотек CUDA/APP, так и при использовании возможностей библиотеки OpenCL. Применение последней дает еще одно немаловажное преимущество – модель GPU-вычислений с использованием OpenCL организована в виде «очереди заданий», что обеспечивает большую отказоустойчивость системы, синхронизируя доступ к разделяемому ресурсу (GPU) из многопоточной среды.

Программа в рамках комплекса, ориентированная на конвейерно-параллельную архитектуру, может использовать различные виды параллелизма, такие как

1) параллелизм независимых ветвей (в рамках одной программы могут быть выделены независимые части программы, не имеющие зависимостей, которые могут исполняться параллельно);

2) параллелизм множества объектов или данных (обработка информации о различных, но однотипных объектах по одному и тому же или почти по одному и тому же алгоритму);

3) параллелизм смежных операций (использование принципа конвейеризации вычислительных операций).

В рамках исследования был разработан специальный графический интерфейс визуально-блочного программирования, позволяющий визуализировать процесс конструирования и редактирования схем решения задач на этапе проектирования. Модуль визуализации, входящий в состав графического интерфейса пользователя, помимо непосредственной выдачи результатов счета может быть снабжен когнитивными дополнениями.

Работа ПК основывается на подсистеме диспетчеризации команд, основной задачей которой является управление процессом передачи данных между модулями и инициации их запуска, путем передачи вычислителям управляющих команд. Общий алгоритм работы системы диспетчеризации можно описать следующим образом:

1) построение начального состояния (подготовка множества команд инициализации вычислителей и команд запуска модулей-«поставщиков» ресурсов, загрузка модулей, необходимых для решения прикладной задачи);

2) построение списка применимых команд;

3) переход к п. 7 в случае отсутствия применимых команд;

4) выбор применимой команды, обладающей наивысшим приоритетом (приоритет команды прямо зависит от количества аппаратных ресурсов, требуемых для ее исполнения, приоритетный выбор наиболее ресурсоемкой команды в каждом состоянии максимизирует нагрузку на узлы КВУ);

5) передача выбранной приоритетной команды вычислителю;

6) обновление внутреннего состояния комплекса, далее переход к п. 2;

7) при наличии незавершенных команд, переданных вычислителям:

а) ожидание подтверждения от вычислителей;

б) обновление внутреннего состояния ПК, далее переход к п. 2;

8) завершение работы.

Предложенный подход к диспетчеризации работы, основанный на «жадном» алгоритме локально-оптимального выбора применимых команд, несмотря на его простоту, является достаточно эффективным решением, позволяющим задать ограничения на количество используемых аппаратных ресурсов кластера.

2. Графический интерфейс визуального программирования

Интерфейс реализует поддержку функций универсальной моделирующей среды, обеспечивает возможность выбора предобработки, определение типа и конфигурации нейронной сети, сохранение и визуализация результатов. При этом последовательность действий, приводящая к желаемому результату – минимальна. Обеспечивается максимальная гибкость формирования решаемых задач.

Общая функциональность, доступная из любой области интерфейса, – это выпадающий список выбора схемы задания, индикатор статуса корректности синтаксиса схемы (корректная, некорректная, непроверенная), кнопки добавления, сохранения, удаления, запуска на выполнение и остановки запущенного задания, кнопки навигации по состоянию схемы во времени, расположенные в верхней части интерфейса (рис. 1). Форма работы с входными и выходными данными (“Read and Save modules”) предназначена для конфигурирования тех включенных в схему задачи модулей, что обладают только входными или только выходными каналами.

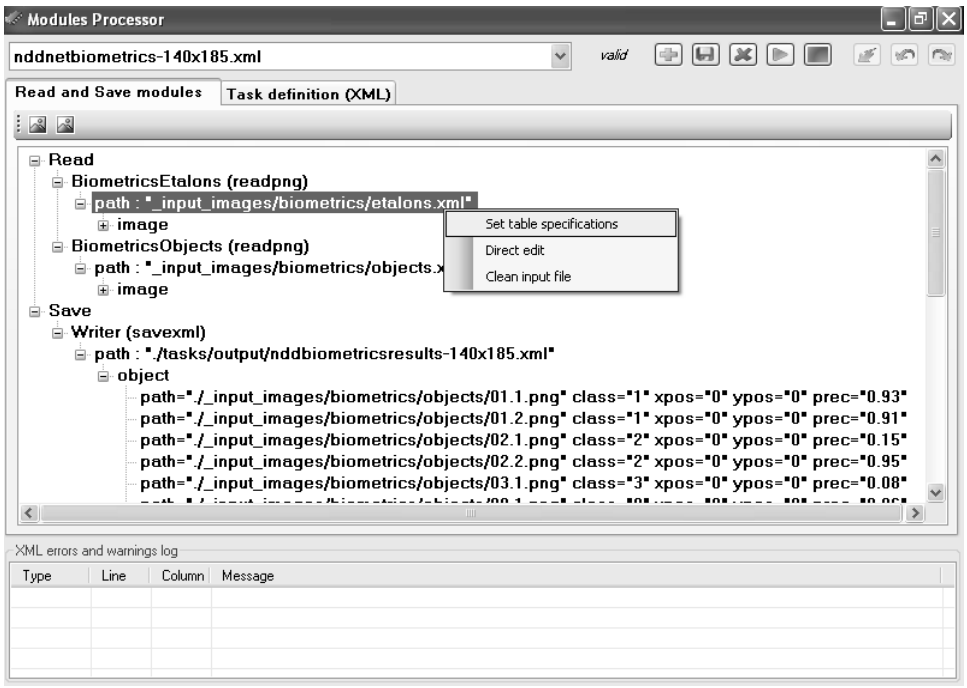

Рис. 1 – Форма работы с входными и выходными данными

В первом случае это модули, загружающие данные для решения поставленной задачи (подключаются к корню “Read” дерева данных), во втором это модули, сохраняющие результаты этой работы (подключаются к корню “Save” дерева данных). Источниками данных могут выступать как xml-файлы, так и любые другие файлы и/или значения параметров модулей.

Форма работы с текстом задачи (“Task definition (XML)”), представленная на рис. 2, предназначена как для определения новых схем заданий, так и редактирования существующих. Она разделена на две части: дерево доступных модулей и область представления задачи. Дерево доступных модулей содержит описания модулей, включающие информацию о доступных параметрах модулей и их умолчательных значениях, а также информацию о доступных каналах. Форма представления задания в виде функциональных блоков (рис. 2) позволяет производить редактирование схемы задачи с использованием функциональных блоков, перетаскиваемых на рабочее пространство помощью drag-and-drop-операций манипулятором типа «мышь». Связи между блоками (каналы) устанавливаются с помощью стрелок. Каждый модуль отображается как прямоугольник, окруженный доступными для подключений каналов областями. Входные каналы сгруппированы над прямоугольником модуля, а выходные под ним.

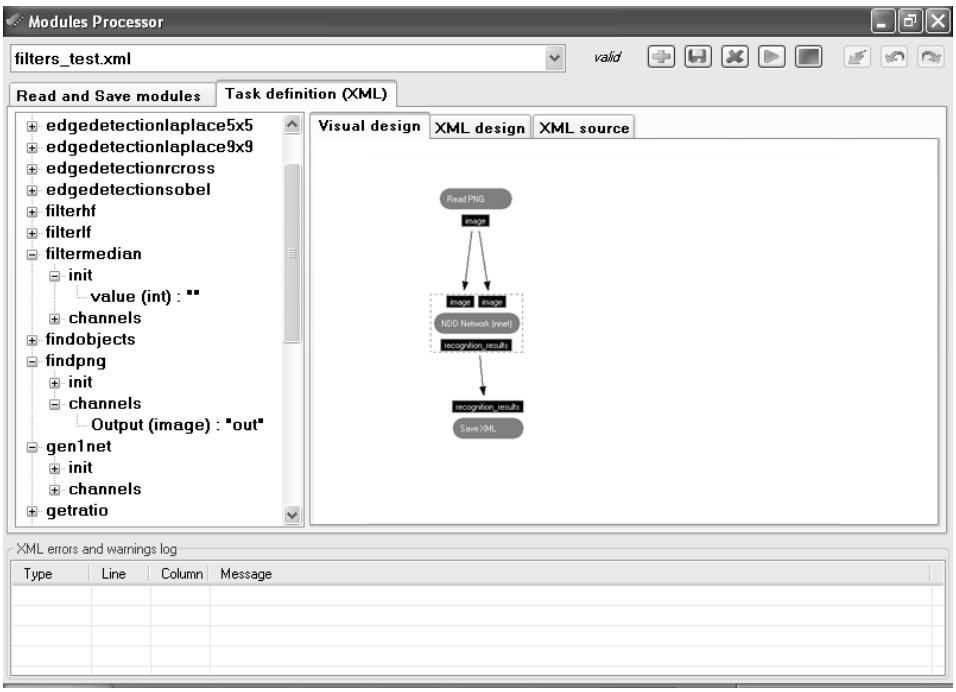

Рис. 2 – Форма представления задания в виде функциональных блоков

Сгенерированное описание прикладной задачи в формате xml доступно для просмотра и редактирования на вкладке ”XML source”.

3. Пример схемы обработки информации

В качестве примеров применения разрабатываемого программного комплекса в том числе были исследованы и успешно решены задачи

1) поиска и распознавания изображений летательных аппаратов на аэрокосмическом снимке;
2) «закраски» регионов на снимках дистанционного зондирования;
3) фильтрации изображений с учителем [11];
4) сжатия изображений с потерями;
5) контроля и диагностики сложных технических систем;
6) распознавания формул в отсканированных математических статьях [12].

Рассмотрим более подробно схему решения задачи поиска и распознавания изображений летательных аппаратов на аэрокосмическом снимке с использованием алгоритмов удаления фона, выделения и склейки объектов и искусственной нейронной сети Хемминга. При решении данной тестовой задачи был использован следующий набор модулей:

- модули чтения данных (readpng, findpng);
- непараллельные фильтры (autorotate, resize, ratiocount, scanwindowfilter);
- параллельные фильтры (magicwand, smalldel, findobjects, glueobjects);
- реализация ИНС (hemming);
- модули сохранения данных (savexml, savepng);
- модуль визуализации результатов поиска целевых объектов (scanwindowmark).

Алгоритм обработки входных данных можно описать следующим образом (следует отметить, что, хотя описание дается в виде четырех независимых этапов, во время работы комплекса они выполняются в конвейерно-параллельном режиме):

1. Чтение и предобработка исходных данных (эталонных изображений) и обучение ИНС:

- чтение эталонных изображений;
- разворот эталонных изображений относительно их линии положения;
- изменение размеров эталонных изображений (подготовка данных для ИНС);
- обучение ИНС на подготовленной базе изображений.

2. Вычисление геометрических особенностей, характерных для эталонных изображений (например, отношение площади объекта к подложке).
3. Поиск и распознавание объектов:

- чтение изображений (далее – карт), на которых необходимо производить поиск объектов;
- удаление фона на карте;
- удаление мелких объектов на карте;
- выделение объектов или их фрагментов на карте;
- склейка выделенных объектов (фрагментов), предварительная обработка найденных объектов – отсев ложных объектов на основе сопоставления геометрических особенностей;
- разворот найденных изображений относительно их линии положения;
- изменение размеров найденных изображений;
- распознавание каждого из найденных изображений с формированием структур, описывающих результаты распознавания;
- фильтрация результатов распознавания ИНС с удалением результатов с малой вероятностью распознавания и возможных дубликатов;
- сохранение результатов распознавания в файл формата xml.

4. Маркировка найденных объектов на копиях карт и сохранение изображений.

В качестве входных данных использовался набор из 142 эталонных изображений летательных аппаратов, разделенных на 11 классов, пример сохраненного результирующего изображения приведен на рис. 3.

Рис. 3 – Обнаружение целевых объектов

При тестировании системы на КВУ (10 тестовых запусков на каждый эксперимент) были получены результаты, представленные в следующей таблице:

Таблица – Результаты тестирования производительности (ускорение)

		Кол-во вычислительных потоков на узел							
		1	2	3	4	5	6	7	8
Кол-во узлов КВУ	1	1.00	1.82	2.71	3.05	3.10	3.04	3.04	3.03
	2	1.21	3.26	3.30	4.85	4.62	4.74	4.59	4.68
	3	1.25	3.72	3.76	5.88	5.22	5.24	5.12	5.06
	4	1.25	2.64	3.34	3.86	5.30	4.94	4.07	5.78
	5	1.22	3.62	3.80	4.12	4.84	3.66	4.91	5.33

Заключение

Разработанный комплекс инструментальных программных средств для проектирования нейросетевых прикладных систем, включает в себя: программные средства преобразования поступающих данных в специально предназначенную для последующего анализа форму; блок анализа данных на основе нейросетевых методов, предназначенный для обработки поступающих данных; универсальный графический интерфейс проектирования нейросетевых прикладных систем с помощью визуально-блочного программирования. Предложенная архитектура позволяет полноценно использовать в том числе и ресурсы гетерогенной среды GPU-кластера. Полученные результаты соответствуют заявленной проблеме и вносят вклад в ее решение, отличаются новизной и оригинальностью.

Список литературы

1. Kepler official website. URL: https://kepler-project.org/ (дата обращения: 10.07.2013).
2. *Altintas I., Berkley C., Jaeger E., Jones M., Ludascher B., Mock S.* Kepler: an extensible system for design and execution of scientific workflows. In Proceedings of 16th International Conference on Scientific and Statistical Database Management, pp. 423-424, 2004.
3. ANSYS official website. URL: http://www.ansys.com/ (дата обращения: 10.07.2013).
4. *Шукла Д., Шмидт Б.* Основы Windows Workflow Foundation. – М.: ДМК Пресс, 2008.
5. CODE official website. URL: http://www.cs.utexas.edu/users/code/ (дата обращения: 31.01.2013).
6. Суперкомпьютерная программа «СКИФ» Союзного государства. URL: http://skif.pereslavl.ru/skif/ (дата обращения: 10.07.2013).
7. xianyi/OpenBLAS Wiki. URL: https://github.com/xianyi/OpenBLAS/wiki (дата обращения: 10.07.2013).
8. OpenCL official site. URL: www.khronos.org/opencl/ (дата обращения: 10.07.2013).
9. NVIDIA CUDA official Site. URL: http://www.nvidia.com/object/cuda_home_new.html (дата обращения: 10.07.2013).
10. *Талалаев А.А.* Организация конвейерно-параллельных вычислений для обработки потоков данных. – Информационные технологии и вычислительные системы, №1, 2011, с.8-13.
11. *Хачумов В.М., Фраленко В.П.* Высокопроизводительная обработка изображений на кластерных устройствах. – Нейрокомпьютеры: разработка и применение, № 6, 2012, с.38-45.
12. *Кирюшина А.Е.* Распознавание математических символов с использованием сверточной нейронной сети – Труды XVII Молодежной научно-практической конференции «Наукоемкие информационные технологии» SIT-2013 (апрель 2013). – УГП им. А.К. Айламазяна, 2013.