УДК 004.272.43+004.272.32

МОДЕЛИРОВАНИЕ АРХИТЕКТУРЫ ГРАФОДИНАМИЧЕСКОЙ МАШИНЫ НА БАЗЕ ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА

М.М. ТАТУР

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 28 апреля 2015

Разработана модель параллельной архитектуры проблемно-ориентированного процессора. С целью верификации полученных технических решений проведено моделирование архитектуры на тестовых примерах на базе кластера GPU с использованием технологии CUDA. В результате модель позволила сымитировать отдельные блоки и исследовать вычислительные процессы, которые протекают в исследуемой архитектуре.

Ключевые слова: ассоциативный поиск, семантическая сеть, граф, параллельные вычисления.

Введение

Интеллектуальная обработка информации неразрывно связана с такими научными областями, как обработка сигналов и изображений, распознавание образов, семантическая обработка знаний (методы искусственного интеллекта). Границы этих областей достаточно условны и стираются при создании конкретных прикладных информационных систем. Становится все более очевидным, что интеллектуальный анализ информации в каждом конкретном случае представляет собой последовательность различных, порой достаточно сложных, процедур обработки. Возможно, поэтому ни один отдельно взятый математический аппарат (математическая статистика, нейронные сети, нечеткие множества, семантические сети и др.) не в состоянии эффективно решать все многообразие интеллектуальных задач.

Более того, алгоритмы интеллектуальных вычислений носят переборный или комбинаторный характер, т.е. сложность и время вычисления задач возрастает нелинейно с ростом числа переменных. Подобные алгоритмы нередко включают в контуры управляющих систем реального времени, к которым предъявляются особо высокие требования по производительности. Как следствие, становится актуальной проблема эффективного распараллеливания вычислений и их реализации на суперЭВМ, вычислительных кластерах или проблемно-ориентированных компьютерах с оригинальными архитектурами, т.е. создания эффективных высокопроизводительных специальных средств вычислений.

С одной стороны, мы имеем прикладную задачу, а со второй стороны – некоторую аппаратную платформу, на которой эта задача может быть решена. Процесс проектирования программно-аппаратной интеллектуальной системы состоит в выборе как самой аппаратной платформы, так и разработке всего комплекса моделей, алгоритмов и, в конечном счете, программ для данной аппаратной платформы. Очевидно, что процесс проектирования неформален и содержит неограниченное число возможных технических реализаций, отличающихся технико-экономическими, конструктивными параметрами и эффективностью.

Современные унифицированные параллельные компьютеры (GPU, суперЭВМ и вычислительные кластеры, облачные вычислительные сети) не дают принципиального решения проблемы, т.к. не указывают пути распараллеливания сложного когнитивного вычислительного процесса на относительно простую регулярную параллельную архитектуру. Практика

свидетельствует о том, что прикладная задача на кластере или другом параллельном компьютере чаще всего получает экстенсивное ускорение вычислений. Еще хуже обстоит дело со специализированными вычислителями, время и стоимость создания которых велики, а унификация в решении прикладных интеллектуальных задач стремится к нулю. В этом контексте надо осторожно относиться к оценке «перспективности» современных нейронных, семантических и ассоциативных компьютеров. Зачастую они имеют либо урезанный вариант SIMD-архитектуры и не гарантируют эффективности решения прикладной задачи, либо представляют собой узкоспециализированный спецпроцессор.

Ниже приведены результаты архитектурного проектирования и модельной верификации. Сформулированы объективные ограничения и некоторые закономерности распараллеливания алгоритмов обработки графов.

Архитектура моделируемого процессора

В ходе предварительных исследований была предложена архитектура параллельного процессора (рис. 1), проблемно-ориентированного на класс задач обработки графов (графо-динамическая машина) [1–3]. Архитектура процессора относится к SIMD-типу и представляет собой независимый блок ассоциативной памяти большого объема с соответствующим интерфейсом доступа. Система команд процессора содержит команды чтения и записи данных с адресацией ячеек по их содержимому. В качестве тэга (поискового признака) может быть использован любой набор бит данных, настройка выполняется посредством самой команды чтения/записи. Операция записи может выполняться одновременно в любое количество ячеек памяти (от нуля до всего объема памяти в зависимости от поискового запроса), считывание возможно только последовательно по одной ячейке за одну операцию (основное узкое место архитектуры).

Основными отличительными характеристиками архитектуры являются простота наращивания числа процессорных элементов (ПЭ) и объема памяти процессора. Логическая схема ПЭ чрезвычайно простая, что позволяет реализовать огромное их количество на одном кристалле, в соответствии с чем снижается стоимость одной ячейки памяти.

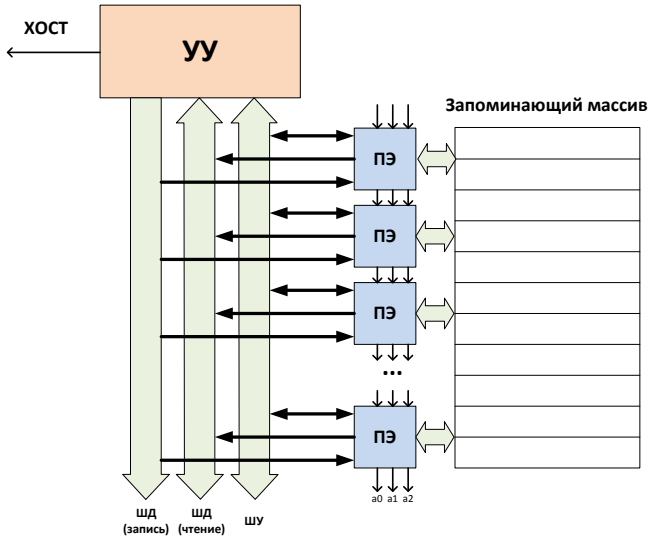

Рис. 1. Архитектура графодинамической машины

В ходе исследования данная архитектура была реализована в виде программной модели на базе вычислительного кластера и технологий (NVidia CUDA). Это позволило провести модельные эксперименты для верификации разработанной архитектуры процессора, системы команд процессора и предложенной схемы обработки информации в целом.

Постановка модельных экспериментов и их обсуждение

Программная модель реализована на базе вычислительного кластера, состоящего из управляющего узла и 7 вычислительных узлов. Характеристики узлов представлены в табл. 1.

Таблица 1. **Аппаратные ресурсы кластера, используемого для моделирования параллельной архитектуры**

Управляющий узел	Вычислительный узел (x7)
Blade: GPU SuperBlade SBI-7127RG 2 x CPU Intel Xeon E5606 24 Gb RAM 2x SSD 80Gb 4x HDD 300Gb InfiniBand 4x QDR (40Gbps) Network 2x Gigabit Ethernet	Blade: GPU SuperBlade SBI-7127RG 2 x CPU Intel Xeon E5-2650 32 Gb RAM 2x Tesla M2075 6 Gb RAM InfiniBand 4x QDR (40Gbps) Network 2x Gigabit Ethernet

Для обмена данными между вычислительными узлами используется MPICH2 (стандарт интерфейса обмена данными MPI, Message Passing Interface). Планирование задач и общее управление ресурсами вычислительного кластера осуществляется посредством TORQUE Resource Manager. Основной рабочий проект реализован на C++, сборка проекта на тестовых машинах осуществляется в MVS 2010, на кластере – посредством Makefile. Проект находится в открытом доступе на github.

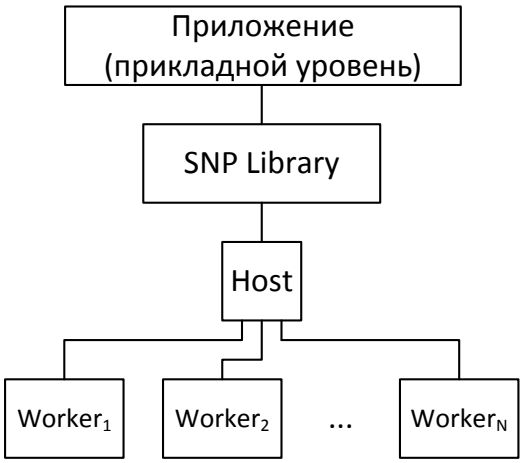

Рис. 2. Структура проекта

Структура программного обеспечения представляет собой несколько логически независимых модулей.

1. Собственно приложение. Как правило, прикладная программа с удобным графическим пользовательским интерфейсом и возможностью работы в интерактивном режиме.

2. SNP Library – разработанная в рамках проекта библиотека для интеграции в произвольные прикладные проекты. Предоставляет пользователю интерфейс, имитирующий систему команд процессора и скрывает все взаимодействие с вычислительным кластером. Модуль интегрируется в приложение на стадии сборки исполняемого файла, вся необходимая внешняя настройка осуществляется посредством конфигурационного файла: задается IP-адрес и порт, по которому идет общение с кластером, авторизационные данные. Общение с вычислительным кластером (в частности хост-процессом) происходит по сети Ethernet.

Для запуска вычислительной системы SNP Library с помощью внешних скриптов формирует на управляющем узле кластера служебные файлы с конфигурацией системы (необходимое количество вычислительных узлов, требуемый размер свободной памяти на винчестере и т.п.) и передает управление TORQUE. В свою очередь, TORQUE запускает на вычислительных узлах необходимые служебные процессы (Host, Worker).

3. Host – служебный процесс, запущенный на одном из вычислительных узлов кластера, который берет на себя ответственность за взаимодействие с SNP Library по сети Ethernet. Принимает и обрабатывает пакеты с командами процессору, рассылает команду всем рабочим процессам (Worker), ожидает ее исполнения, собирает, агрегирует и отправляет результат обратно головному процессу (SNP Library).

4. Worker – служебный процесс, запущенный по одному экземпляру на каждом из вычислительных узлов кластера. Принимает команды от хост-процесса (посредством MPI),

исполняет их на имеющихся графических процессорах (посредством NVidia CUDA), обрабатывает и отправляет результат обратно хост-процессу.

Порядок проведения эксперимента.

Тестовая задача включает в себя следующую последовательность действий.

1. Параллельная запись во все ячейки памяти фиксированного числового значения.
2. Поисковый запрос по тому же числовому значению.
3. Дальнейшая последовательная вычитка результата запроса (соответствует всему объему памяти процессора).

Тесты были выполнены для различных конфигураций процессора согласно табл. 2.

Таблица 2. **Данные конфигураций процессора при проведении экспериментов: [A] (C; M; N), где A – количество вычислительных узлов, C – размер ячейки памяти (Бт), M – количество ячеек в одном ПЭ (размер локальной памяти ПЭ), N – количество ПЭ**

Тест 1. Переменное число вычислительных узлов	Тест 2. Переменное число ПЭ	Тест 3. Переменное число ячеек в одном ПЭ
[1] (4; 1; 28627)	[3] (4; 1; 86016)	[3] (4; 1; 86016)
[2] (4; 1; 57344)	[3] (4; 1; 172032)	[3] (4; 2; 86016)
[3] (4; 1; 86016)	[3] (4; 1; 258048)	[3] (4; 4; 86016)
[4] (4; 1; 114688)	[3] (4; 1; 344064)	[3] (4; 8; 86016)
[5] (4; 1; 143360)	[3] (4; 1; 430080)	[3] (4; 16; 86016)
[6] (4; 1; 172032)	[3] (4; 1; 516096)	

В ходе экспериментов было измерено среднее время выполнения инструкции, а также распределение временных затрат между операциями чтения, записи и временем простоя. Некоторые результаты исследований, а также пример прикладной задачи, опубликованы в [4].

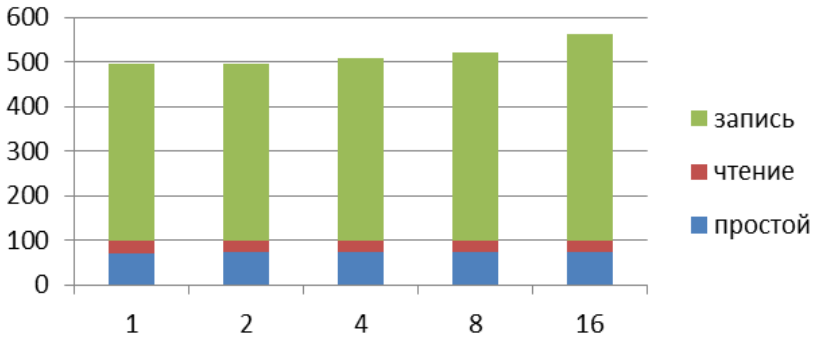

Рис. 3. Среднее время выполнения инструкции (тест 3)

Заключение

Разработанная программная модель позволяет верифицировать работоспособность предложенной архитектуры процессора и схемы семантической обработки данных в целом. Тем не менее, для практического применения необходимо выполнить ряд работ по совершенствованию ПО. В дальнейшем планируется:

– оптимизировать логику работы рабочего процесса (Worker): добавить многопоточную работу с графическими процессорами, улучшить алгоритм распределения данных между узлами и графическими процессорами, провести поиск узких момент в работе с NVidia CUDA и т.п.;

– уменьшить долю времени выполнения, затрачиваемую на передачу данных по сети;

– улучшить (упростить) процесс разворачивания вычислительной системы;

– разработать пользовательский интерфейс для взаимодействия с моделью в интерактивном режиме;

– реализовать (согласно схеме на рис. 2) программную модель процессора в нотации «ключ-значение» для проведения сравнительных экспериментов;

– реализовать типовые задачи на новом API (интерфейсе SNP Library), провести модельные финальные эксперименты.

MODELING OF THE GRAF-DYNAMIC MACHINE ARCHITECTURE WITH COMPUTING CLUSTER

M.M. TATUR

Abstract

The model of the parallel architecture of problem-oriented processor has been developed. For the purpose of verification of delivered technical solutions the simulation of the architecture on the test cases with cluster GPU using CUDA technology has been provided. As a result, the model allowed to simulate of individual units and to explore the computational processes that occur in the studied architecture.

Список литературы

1. *Голенков В.А., Гулякина Н.А.* / Матер. 2-й Междунар. науч. техн. конф. «Открытые семантические технологии проектирования интеллектуальных систем» Минск, 2012. С. 23–52.
2. *Verenik N., Girel A., Seitkulov Y. et. al.* // Proc. of 10-th Int. Conf. on Digital Technologies. Slovakia, 2014. P. 367–371.
3. *Verenik N., Seitkulov Y., Girel A. et. al.* // Eurasian journal of mathematical and computer application. 2014. Vol. 2, Iss. 2. P. 92–101.
4. *Ивашенко В.П., Вереник Н.Л., Гирель А.И. и др.* // Матер. 5-й Междунар. конф. «Открытые семантические технологии проектирования интеллектуальных систем». Минск, 2015. С.133–140.