# Об использовании современных многоядерных процессоров для решения математических и прикладных задач в нейросетевом логическом базисе

**Минип Аляутдинов, Илья Воронков, Дмитрий Пантюхин, Павел Скрибцов (Москва)**

**В статье рассмотрены современные графические и многоядерные микропроцессоры с параллельной архитектурой G80 nVidia, CELL (IBM) и Intel Core с точки зрения их эффективного применения в масштабируемых нейрокомпьютерных конфигурациях. Приведены результаты, полученные в рамках проекта по разработке пакета программ для решения плотных систем линейных алгебраических уравнений и задач газовой динамики. Описаны методы программно-аппаратного моделирования нейросетевых вычислений на базе графических плат nVidia (G7), а также на базе кластеров, оснащённых такими платами, и кластеров из обычных компьютеров, связанных локальной сетью.**

## Введение

Проблема создания эффективных масштабируемых нейрокомпьютерных конфигураций является весьма актуальной в связи с расширением области применения нейросетевых технологий для решения задач, требующих интенсивных вычислений в реальном масштабе времени. К таким задачам относятся обработка сигналов и изображений, потоковой мультимедийной и геофизической информации.

В настоящее время реализация нейросетевых алгоритмов осуществляется программной, программно-аппаратной эмуляцией и чисто аппаратными средствами [1]. Программно-аппаратные эмуляции выполняются на базе компьютеров с широкомасштабным параллелизмом. Чисто аппаратными средствами реализации нейронных сетей (НС) являются многопроцессорные нейроподобные СБИС, ПЛИС и нейроИС. Аппаратная реализация НС-алгоритмов является самым быстродействующим, но дорогостоящим вариантом.

На сегодняшний день увеличение вычислительной мощности одноядерных процессоров за счёт повышения тактовой частоты и архитектурных усовершенствований практически нерентабельно [2]. Ведущие производители микропроцессоров переходят на разработку многоядерных приборов с новой архитектурой, обеспечивающей распараллеливание обработки данных.

Появление многоядерных процессоров является качественным скачком на пути создания эффективных супервычислителей, обладающих существенно более высокими показателями производительность/стоимость по сравнению с существующими системами на базе суперЭВМ и кластерных систем. Использование многоядерных процессоров предоставляет гибкие возможности в части изменения конфигураций и масштабирования мощности вычислительных систем, – от персональных компьютеров и рабочих станций до серверов и кластерных систем [3].

Теоретически многоядерные процессоры в силу своих исключительных вычислительных возможностей являются наиболее перспективными средствами аппаратной поддержки нейросетевых и информационных технологий, связанных с интенсивными вычислениями.

Среди многоядерных процессоров с параллельной архитектурой в настоящее время наиболее известны графические процессоры (Graphics Processing Unit, GPU) и центральные процессоры (CPU) типа IBM CELL и Intel Core. В течение последнего десятилетия наиболее динамично развивались GPU, что было обусловлено требованиями компьютерной графики к повышению вычислительной мощности графических плат, необходимой для построения высококачественных изображений в реальном масштабе времени. Первые образцы многоядерных микропроцессоров IBM CELL и Intel Core появились только в 2006 г. Характеристики некоторых многоядерных процессоров приведены в таблице 1 [3].

**Таблица 1. Основные характеристик современных многоядерных процессоров**

| Процессор | CPU | GPU | Cell | |
|---|---|---|---|---|
| | Dual-Core Xeon 5160 | ATI 1900XT | nVidia n8800 GTX | IBM Cell |
| Объем памяти, Гб | 32 | 1 | 0,768 (GDDR3) | 0, 512 |
| Скорость обмена с памятью, Гб/с | 6 | 50 | 86,4 (DDR3) | 26 |
| Пиковая производительность, GFLOPS | 48 | 360 | 360...520 | 256 |

## Современные многоядерные процессоры

### Графические микропроцессоры с параллельной архитектурой

В течение последних 20 лет архитектура GPU базировалась на традиционном графическом конвейере, который состоял из последовательных этапов обработки потока графичес-

ких данных: вертексной (обработки вершин), сборки вершин в треугольники, формировании пиксельных фрагментов, обработки на уровне пикселей, растеризации и построении кадра изображения. Блоки первых поколений GPU имели жёсткую структуру, ограниченную функциональность и были не программируемыми. Такая архитектура обладала рядом недостатков, связанных со специализацией конвейерных узлов и ограничениями по типу данных и составу команд, точности вычислений, ресурсам (регистры, текстуры, память), невозможностью повторного использования в процессе обработки элементов потока данных, различием аппаратных характеристик у различных производителей GPU, несовершенством механизмов балансирования загрузки их процессоров. Это приводило к неэффективному использованию вычислительного потенциала аппаратных ресурсов GPU в целом.

Графические процессоры с модифицированной архитектурой, выпускаемые в последние годы, уже не имели некоторых из перечисленных недостатков, обеспечивали возможность программирования наиболее важных узлов конвейера (вертексных и пиксельных) и существенно расширили масштабирование аппаратного параллелизма. Современные GPU содержат полностью программируемые параллельные геометрические и пиксельные процессоры, снабжённые полным набором команд для выполнения арифметических и логических операций с поддержкой 32-разрядного формата векторных и скалярных операций с плавающей точкой. Для быстрой обработки больших графических наборов данных (вершин и фрагментов) в них используется потоковая модель обработки с параллелизмом.

Такие GPU стали привлекательными для реализации неграфических вычислений (GPGPU – General Purpose GPU), что стимулировалось двумя основными факторами: критерием производительность/стоимость и темпами роста производительности GPU, которая удваивалась каждые 6 месяцев. (Производительность CPU в среднем удваивалась каждые 18 месяцев.)

Используя массивный параллелизм и векторные процессоры, современные графические устройства способны исполнять многие из приложений, ранее реализованных на векторных (SIMD) суперкомпьютерах. В настоящее время сфера использования GPU расширяется благодаря возможности их программирования на языках высокого уровня. Сегодня на GPU эффективно реализованы: задачи физического моделирования, операции линейной алгебры, решение дифференциальных уравнений в частных производных, обработка сигналов и изображений, нейросетевая обработка и др.

В конце 2006 г. компанией NVIDIA был выпущен графический процессор нового поколения GeForce 8800 (G80) [4]. При разработке этого процессора были пересмотрены и существенно переработаны проектные решения и архитектуры. Помимо усовершенствований, связанных с разработкой более производительного GPU с улучшенным качеством изображения, было выдвинуто требования обеспечения интенсивных вычислений с плавающей точкой для реализации различных неграфических приложений.

Процессор NVIDIA GeForce 8800 (G80) является многоядерным и многопоточным высокопроизводительным микропроцессором. По своим функциональным характеристикам и вычислительной мощности он может рассматриваться и как графический процессор, и как универсальный процессор для эффективной реализации неграфических приложений, требующих интенсивных вычислений. В качестве графического процессора он полностью реализует функции классического конвейера. В качестве универсального процессора, на операциях с плавающей точкой он превосходит по критерию производительность/стоимость все существующие традиционные и многоядерные CPU и GPU.

Базовыми инновациями, использованными в G80, являются:

- унифицированная архитектура массива ядерных потоковых процессоров с плавающей точкой, пригодных для исполнения как графических конвейерных операций (геометрических преобразований, обработки вершин и пикселов), реализуемых единообразно на потоковых процессорах, так и неграфических вычислений;
- технология NVIDIA GigaThread Technology – широкомасштабная многопоточная архитектура, поддерживающая исполнение тысячи независимых, параллельно исполняемых нитей (потоков команд), обеспечивающая высокую эффективность обработки потоковых данных и использования вычислительного потенциала многоядерных GPU. (Для сравнения, современные многоядерные CPU поддерживают работу на один-два порядка меньшего количества нитей.)

Кроме того, видеоплаты на базе G80 поддерживают SLI-технологию, обеспечивающую параллельную работу нескольких GPU.

Основные характеристики процессора G80: технология 90 нм; 681 млн. транзисторов; унифицированная архитектура в виде массива 128 скалярных 32-битных АЛУ (потоковых процессоров, SP) с плавающей точкой (IEE 754); 384-разрядная шина памяти; 6 независимых контроллеров памяти шириной 64 бита, поддержка GDDR4 (1,8 ГГц); частота ядра до 575 МГц (G80 GTX). Каждый потоковый процессор GeForce 8800 GTX работает на тактовой частоте 1,35 ГГц и поддерживает сдвоенную обработку скалярных операций MAD и MUL (операции накопления), что позволяет достичь производительности порядка 520 GFLOPS.

Помимо указанных компонентов, процессор G80 содержит дополнительные аппаратные ресурсы, необходимые для выполнения текстурных, растровых и других операций графического конвейера. В перспективе компания NVIDIA может выпустить на основе G80 универсальные многоядерные процессоры для неграфических приложений с отключенными графическими компонентами.

При реализации на процессоре G80 неграфических вычислений наиболее важными являются унифицированные потоковые процессоры, доступные им ресурсы памяти, коммуникационные и управляющие средства. На рисунке 1 приведена блок-схема унифицированного массива процессоров G80.

Процессор G80 содержит 128 потоковых процессоров (SP), организованных в 8 групп по 16 процессоров. Потоковые процессоры являются унифицированными скалярными процессорами с плавающей точкой, обрабатывающими не только графические, но и другие данные. Объединение SP в кластеры позволяет наиболее эффективно использовать аппаратные ресурсы G80: 32-битные регистры, разделяемую внутрикристальную па-

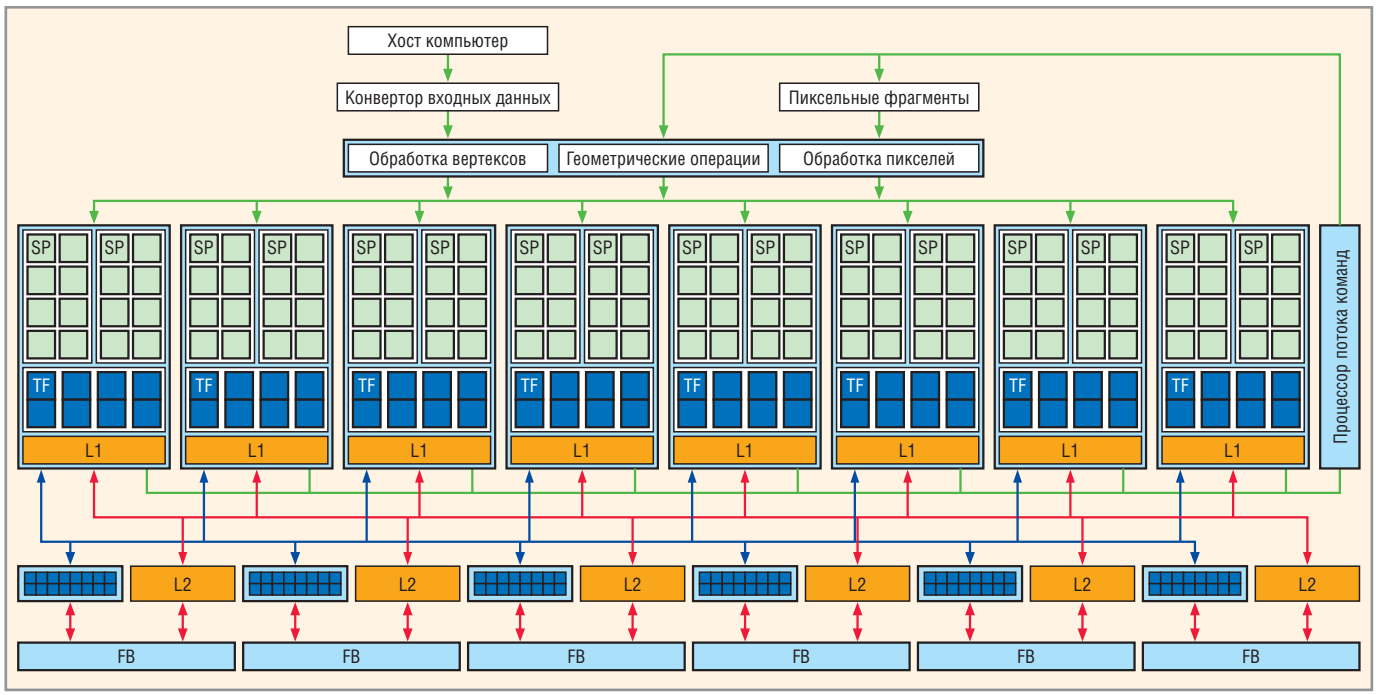


**Рис. 1. Блок-схема унифицированного массива процессоров G80**

мять (16 Кб на кластер), разделяемую кэш-память (64 Кб) для чтения констант из внешней памяти G80, разделяемую кэш-память текстур. Разделяемые ресурсы памяти внутри кластеров позволяют обеспечить синхронизацию и связь между нитями (потоками команд). Каждый потоковый процессор на основе механизмов управления работой нитей способен динамически переназначаться для выполнения конвейерных графических или других вычислительных операций, обеспечивая, таким образом, пиковую загрузку ресурсов GPU и максимальную гибкость при обработке данных.

Кроме массива потоковых процессоров и шейдерных блоков, ориентированных на специфические графические операции, процессор G80 содержит ряд управляющих блоков:

- Host – блок хост-интерфейса, состоящий из буферов приёма команд, вертексных данных и текстур, пересылаемых для обработки GPU центральным процессором;
- Input Assembler принимает исходные данные из памяти системы или локальной памяти, которые поступают извне на входы процессорных блоков, и конвертирует их в формат FP32, параллельно генерируя ряд идентификаторов для разметки повторяющихся операций с вертексами и примитивами;
- Thread Processor управляет загрузкой потоковых процессоров на основе механизма нитей и движением обрабатываемых потоков данных. Например, пиксели или вершины, которые проходят обработку, двигаются по кругу (не кэшируются) с выходов одних мультипроцессорных блоков на входы других;
- Setup/Raster/ZCull расчленяет аппроксимирующие треугольники на пиксели;
- Vertex, Geometry и Pixel Thread Issue запускают на исполнение программы обработки данных различных форматов, готовят данные для шейдерных процессоров в соответствии с их форматом, текущим шейдером и его состоянием, условиями ветвлений и т.д.

В ближайшем будущем управляющие блоки будут унифицированы, чтобы не быть привязанными к конкретным видам графических операций и данных; они превратятся в универсальные блоки, осуществляющие запуск данных на вычисление и преобразование форматов.

Графическая плата с процессором GeForce 8800 обеспечивает суперкомпьютерные возможности. Пиковая производительность плавающей арифметики нового ускорителя превышает производительность любой арифметики, реализованной в доступных сейчас CPU и GPU. Компания NVIDIA приводит (см. таблицу 2) следующие показатели пиковой производительности передовых на сегодня GPU [4]: G80 в сравнении с двухъядерным процессором Intel Core2 Duo (2,66 ГГц) обеспечивает прирост производительности на 1-2 порядка в зависимости от задачи [4].

Унифицированная архитектура G80 обладает гибкостью, достаточной не только для графических приложений, но и для более «серьезных» задач – математического и физического моделирования, распознавания образов, обработки изображений, нейросетевых вычислений и других задач потоковой обработки.

Для эффективной поддержки неграфических приложений для процессоров NVIDIA была создана среда программирования CUDA (Compute Unified Device Architecture) – унифицированная вычислительная архитектура [5] для различных задач, содержащая специальный SDK, API и компилятор Си, которые обеспечивают быструю разработку и адаптацию программ для исполнения на GPU. Среда CUDA пред-

**Таблица 2. Характеристики пиковой производительности современных GPU**

| Графический процессор | G71(nVidia) | R580(ATI) | GeForce 8800(nVidia) |
|---|---|---|---|
| Операции | Vec3+scalar MAD | Vec3+scalar MAD<br>Vec3+scalar ADD | Scalar MAD+ Scalar MUL |
| Число параллельных процессорных элементов | 48 | 48 | 128 |
| Тактовая частота АЛУ, ГГц | 0,65 | 0,65 | 0,35 |
| Пиковая производительность, GFlops (GMuls) | 125 | 125 | 345...520 |

назначена для работы на новом поколении графических процессоров NVIDIA – от G80 и выше. В модели программирования CUDA, GPU рассматривается как вычислительное устройство, способное поддерживать параллельное исполнение большого числа нитей или потоковых программ и являющееся сопроцессором центрального процессора хост-компьютера. На GPU можно перенести интенсивные вычисления, определяя их как функцию, исполняемую на GPU в виде большого числа (около 1000) параллельно работающих нитей.

### Многоядерные микропроцессоры CELL

В феврале 2005 г. компания IBM совместно с компаниями Sony и Toshiba представила прототип нового процессора под кодовым названием CELL. Процессор основан на новой архитектуре компании IBM, которая называется CELLular architecture [6]. Кристалл процессора содержит 64-разрядное управляющее ядро с архитектурой PowerPC и 8 синергетических 32-разрядных векторных процессорных ядер (Synergistic Processing Element, SPE) с SIMD-архитектурой, работающих на частоте порядка 3,2 ГГц (второе поколение процессоров Cell будет работать на частоте 6 ГГц). Процессор способен адресовать до 64 Гб памяти.

Кристалл процессора Cell изготавливается по 90-нм технологии и содержит 234 млн. транзисторов, включая кэш-память L1 объёмом 256 Кб и L2 объёмом 512 Кб. Производительность нового процессора составляет 250 GFlops, что на порядок выше, чем у современных процессоров Intel.

Векторные процессоры соединяются внутренней кольцевой шиной Element Interconnect Bus, к которой также подключена быстродействующая память и контроллеры ввода/вывода. Скорость обработки данных процессором CELL может достигать 16 Гб/с. Его можно применять как одиночный процессор или в составе многопроцессорной системы, поскольку шина ввода/вывода предусматривает возможность прямого соединения с другими процессорами CELL при помощи специального коммутатора.

Первоначально процессор CELL предназначался для применения в игровых приставках Sony Play Station 3. Серверы на базе процессора CELL будут ориентированы на задачи, требующие интенсивных вычислений: графическая визуализация, обработка данных сейсморазведки, обработка космических изображений, шифрование и сжатие данных.

### Многоядерные микропроцессоры Intel Core

В ноябре 2006 г. компания Intel представила первые четырёхъядерные процессоры нового поколения на базе многоядерной процессорной архитектуры Intel Core (Kentsfield) с торговым названием Core 2 Extreme QX6700 [7].

К инновациям, реализованным в архитектуре Intel Core, относятся [8]:

- технология Intel Wide Dynamic Execution, призванная обеспечить выполнение до четырёх команд за каждый такт, повысить эффективность выполнения приложений и сократить энергопотребление;
- технология Intel Intelligent Power Capability, активируя отдельные узлы процессора только по мере необходимости, значительно снижает энергопотребление системы в целом;
- технология Intel Advanced Smart Cache подразумевает наличие общей для всех ядер кэш-памяти L2, совместное использование которой снижает энергопотребление и повышает производительность;
- технология Intel Smart Memory Access повышает производительность системы, сокращая время отклика памяти и оптимизируя таким образом использование пропускной способности подсистемы памяти;
- технология Intel Advanced Digital Media Boost позволяет обрабатывать все 128-разрядные команды SSE, SSE2 и SSE3, широко используемые в мультимедийных и графических приложениях, за один такт.

Процессоры Kentsfield состоят из двух 2-ядерных кристаллов на базе архитектуры Conroe, помещённых на единую подложку. Основные характеристики процессоров Core 2 Extreme QX670: 65-нм техпроцесс; общее число транзисторов 582 млн.; рабочая частота 2,67 ГГц; 1066-МГц системная шина; объём кэш-памяти второго уровня – по 4 Мб на ядро; потребляемая мощность до 120 Вт. Теоретическая пиковая производительность Kentsfield составляет 4 × (10...12) = 40...50 MFlops. На середину 2007 г. был намечен переход на 45-нм техпроцесс, что позволит начать в 2008 г. выпуск однокристальных 8-ядерных процессоров [9].

По заявлениям компании Intel, количество вычислительных ядер в процессорах ежегодно будет удваиваться и достигнет 32 (проект Keifer) к 2010 г. [10]. Каждое ядро Keifer будет обрабатывать одновременно до четырёх потоков (в сумме до 128 потоков). Планируется, что первый кристалл для проекта Keifer будет произведён по 32-нм техпроцессу. Он будет содержать восемь процессорных узлов по четыре ядра в каждом. Каждый узел будет иметь прямой доступ к 3 Мб кэш-памяти последнего уровня (last level cache, LLC) и к 512 Кб кэш-памяти L2 . Все восемь процессорных узлов с кэш-памятью LLC будут объединены кольцевой шиной, что в итоге сформирует 24 Мб кэш-памяти. Процессор Keifer не предполагает работы на высоких частотах, – ожидаемая стартовая частота первых моделей составляет 2 ГГц. Тем не менее, по производительности процессор Keifer должен обойти современный Xeon 5100 в 15 раз. В сентябре 2006 г. компанией Intel был представлен прототип процессора Polaris с 80 ядрами, способный выполнять до триллиона операций в секунду (терапроцессор) [11].

Многоядерные процессоры IBM CELL и Intel сходны по структурным и функциональным характеристикам. Различие между ними состоит в реализации: если компания IBM использует сравнительно традиционные технологии, то фирма Intel – принципиально новые [11]. Например, в процессоре Cell локальная память реализована как упрощённый аналог кэш-памяти первого уровня и является частью кристалла процессора. В терапроцессоре Intel используется технология трёхмерной упаковки оперативной памяти – на подложке строится «сэндвич» из кристаллов оперативной памяти и лежащего над ними кристалла процессора, что позволяет разместить на той же площади больше ядер и подключенной к ним памяти. В ядрах процессоров Cell нет когерентной памяти (хранящей общие для всех ядер данные), в терапроцессоре Intel – есть. В качестве внешнего интерфейса для Cell используется «электронная» технология; для терапроцессора Intel планируется разработать оптический канал с большей в 10 раз пропускной способностью.

Другими инновациями в терапроцессоре Intel являются: транзакционная оперативная память, позволяющая объединить несколько операций чте-

ния/записи в одну транзакцию, для которой гарантируется защита от одновременного чтения/записи со стороны других ядер; усовершенствованные технологии виртуализации, направленные на использование с операционными системами и языками программирования следующих поколений, ориентированными на высоко параллельные вычисления; интеграция высокоскоростных средств ввода/вывода, вплоть до сетевого контроллера, непосредственно на кристалле процессора.

## Программные средства многоядерных процессоров

Любые планы производителей вычислительных средств так или иначе увязаны с планами разработчиков программного обеспечения. В настоящее время не существует универсальных методов программирования многоядерных процессоров. Поскольку использование многоядерных процессоров относительно ново, их средства программирования недостаточно развиты. Методы программирования процессоров GPU и CELL зависят от прикладной ориентации: компьютерная графика, обработка сигналов и мультимедиа. Языки программирования GPU также учитывают различие их архитектурной специфики: тип поддерживаемого параллелизма и иерархию памяти. Каждый язык программирования поддерживает различные формы параллелизма, включающие традиционные и четырёхкомпонентные SIMD-инструкции, различные возможности масштабирования многопоточной обработки (multi-threading), суперскалярных и VLIW-инструкций. Системы памяти процессоров GPU и CELL представляют собой различные комбинации и конфигурации прямого доступа к разделяемой памяти, локальной памяти, кэш-памяти, длинным и коротким векторным регистрам.

### Средства программирования процессоров CELL

На сегодняшний день средства программирования процессоров CELL включают базовые компиляторы для его ядер PowerPC и SPE, библиотеки для обеспечения процессов синхронизации и коммуникации, базовые программы поддержки функций управления (например, загрузки и запуска рабочих программ). Написание прикладных программ для ядер PowerPC и SPE , процедур копирования данных в локальную память SPE (загрузка кодов и данных), распределения вычислений между SPE возлагается на программистов. Однако архитектура процессоров CELL, по сравнению с GPU, является более традиционной и близкой к архитектуре существующих CPU. Следовательно, для программирования CELL могут использоваться уже наработанные технологии компиляторов: группа программистов IBM Research адаптировала компилятор IBM XL для генерации параллельных программ для процессоров CELL из исходных текстов программ [12].

### Средства графического программирования на GPU

Разработчики программ компьютерной графики для современных GPU используют API (OpenGL или Microsoft Direct3D) и драйверы, поставляемые производителями GPU. Эти API ориентированы на графические (shaders) функции GPU. Они похожи и поддерживают две главные функции: компилирование и исполнение графических программ и поддержку библиотеки графических функций, связанных с пиксельной и фрагментной обработкой.

Графические программы пишутся либо на Си-подобных языках типа Cg, HLSL, GLSL, либо на псевдо-ассемблерных языках OpenGL ARB_Fragment_Program и Direct3D Shader Model Assembly. У всех этих языков много общих характеристик: они обеспечивают явную модель параллельного программирования данных, в которой параллелизмом и коммуникациями управляет программист. Это создаёт некоторые проблемы для прикладных программистов, поскольку языки являются низкоуровневыми, явно учитывают характеристики и ограничения GPU, не виртуализируют аппаратные ресурсы и вынуждают программиста изучать аппаратные характеристики, ограничения на максимальный размер программы, получать информацию об ошибках и компиляции и др. Программа, написанная для определённого типа GPU, требует переработки для обеспечения максимального использования вычислительного потенциала GPU другого типа.

### Средства программирования неграфических вычислений на GPU

За последние годы были разработаны различные программные средства для GPU с архитектурой классического графического конвейера. Однако они не получили широкого распространения из-за недостаточного уровня качества, надёжности и интерактивного взаимодействия. Разработанные средства не содержат инструментов управления ошибками, отладкой и профилированием, не обеспечивают требуемую точность вычислений и не поддержаны библиотеками параллельных алгоритмов (например, матричных вычислений). Поскольку описания интерфейсов и машинной системы команд архитектуры GPU обычно не публикуются, приходится использовать графические API для загрузки и выполнения рабочих программ на GPU.

Для разрешения указанных проблем, с целью исполнения на GPU программ неграфического назначения были разработаны языки и системы программирования типа Sh и Brook. Система программирования Sh включает API для языков Си и Си++, библиотеку с динамической генерацией программ и некоторые абстракции для неграфических приложений. Язык и система программирования Brook основаны на потоковой модели программирования и поддерживают параллельный, Си-подобный язык, с некоторыми ограничениями. Язык Brook предназначен для программирования мультимедийных приложений и высокопроизводительных вычислений. Из-за ограничений в языке, вытекающих из выбранной модели программирования, на языке Brook трудно реализовать некоторые алгоритмы с интенсивным обменом данными.

### Средства программирования многоядерных микропроцессоров

Хотя использование многоядерных процессоров может обеспечить значительное увеличение вычислительной мощности, их освоение прикладными программистами может вызвать определённые трудности. В настоящее время разрабатываются инструментальные программно-аппаратные среды с новыми моделями программирования, адекватными параллелизму и поточности обработки данных. К ним относятся системы CUDA (NVIDIA) и PeakStream Platform (PeakStream Inc., PSP) [3]. Новая платформа PSP предназначена для разработки программ для многоядерных процессоров, включая нетрадиционные процессоры, такие как GPU и IBM CELL.

Пакет PSP разработан для приложений с интенсивными вычислениями и предлагает простые в употреблении абстракции, базирующиеся на потоковой модели программирования, которые делают прозрачными для программистов детали реализации различных параллельных аппаратных архитектур и облегчают переносимость программ на эти архитектуры. PeakStream Platform состоит из четырёх главных компонентов: PeakStream API (средства программирования приложений с использованием библиотеки математических функций для многоядерных процессоров); PeakStream VM – виртуальной машины, которая создаёт оптимизированные объектные коды для многоядерных процессоров; PeakStream Profiler, PeakStream Debugger – средств отладки, анализа и оптимизации кода программ. Разработчик использует PSP Си или Си++ API. Эти API реализованы в виде библиотек, которые динамически транслируют API-вызовы к VM в параллельные оптимизированные исполняемые программы. Пакет PSP также включает интерактивные средства отладки и профилирования. PSP-программа, написанная для GPU, будет работать на будущих многоядерных CPU- или CELL-процессорах без переработки и перекомпиляции.

Характеристики PeakStream Platform [3]:

- представление данных в виде 1D- или 2D-потокового массива и автоматическое их распараллеливание;
- поддержка 32- и 64-разрядной точности вычислений;
- использование стандартных Си и Си++ библиотечных функций;
- использование библиотек программ векторно-матричных операций BLAS 1, 2, 3;
- наличие библиотеки решения линейных уравнений;
- наличие генераторов случайных чисел.

Средства программирования:

- языки программирования Си, Си++;
- компиляторы gcc 3.4.5, gcc 4.0.3 или Intel Compiler 9.0;
- отладчик gdb 6.3;
- операционные системы RedHat Enterprise Linux 4.0, update 3.

Для работы PeakStream Platform требуется система с процессором AMD Opteron или Intel Xeon, 1 Гб системной памяти и графической картой на GPU типа ATI R580; последний обеспечивает доступ к аппаратным ресурсам GPU.

### Решение систем линейных алгебраических уравнений

Для реализации одного из методов решения систем линейных алгебраических уравнений на графических платах был выбран язык Brook. Для решения плотных систем линейных алгебраических уравнений был выбран метод бисопряжённых градиентов, позволяющий получать решение для совместных систем. Это итерационный метод, т.е. на каждой итерации получается всё более точное решение системы уравнений. Основным критерием эффективности метода является скорость вычислений, измеряемая в количестве итераций в секунду.

В результате использования графической платы в качестве вычислительной платформы удалось повысить скорость решения системы алгебраических уравнений в 3-4 раза (в зависимости от размерности решаемой системы) по сравнению с реализацией того же алгоритма на CPU. Допустимая размерность задачи до 1024 × 1024 (матрица системы полностью помещается в памяти графической платы). Было также отмечено, что реализация данного метода на GPU пригодна только в случае хорошо обусловленных систем уравнений, поскольку отсутствие поддержки вычислений с двойной точностью и неточное выполнение вычислений с одинарной точностью приводит к накоплению ошибок и может привести к неустойчивости метода бисопряжённых градиентов. Для хорошо обусловленных систем указанные недостатки GPU-вычислений приводят к увеличению количества необходимых итераций с целью получения решения с заданной точностью.

Таким образом, в области задач линейной алгебры применение GPU позволяет значительно повысить скорость вычислений. Для повышения точности вычислений возможно применение специальных алгоритмов.

### Решение задач газовой динамики

Для задач газовой динамики, реализации на графических платах и на кластерах отличаются только способом распараллеливания нейросетевого алгоритма. Этот алгоритм основан на методе крупных частиц – одной из разновидностей методов частиц, широко используемой в современных исследованиях. Разработка метода крупных частиц проводилась О.М. Белоцерковским и Ю.М. Давыдовым в ВЦ АН СССР, начиная с 1965 г. [18] и явилась развитием идей метода частиц в ячейках (PIC) Ф. Харлоу.

Авторами была использована клеточная нейронная сеть, для которой были аналитически выбраны весовые коэффициенты на основе априорных данных о том, как из входных сигналов вычисляются выходные. Если отвлечься от физики и выбранного нейросетевого алгоритма, то для численного решения системы нелинейных нестационарных уравнений динамики вязкого, сжимаемого, теплопроводного газа при наличии диффузии и химических реакций, вычислительный алгоритм устроен следующим образом.

Пусть имеется трёхмерный объём газа, представляющий собой массив структур:

```
struct PWNGasCell {
   float u; // x-компонента скорости в ячейке
   float v; // y-компонента скорости
   float w; // z-компонента скорости
   float e; // внутренняя энергия
   float p; // давление
   float ro; // плотность
   float gamma; // показатель адиабаты
   float type; // тип ячейки (газ, постоянный газ, стенка, твердое тело, и т.п.)
};
```

В каждый момент времени необходимо вычислить для каждой ячейки новые компоненты структуры, которые зависят от компонентов на предыдущем временном слое. При этом в расчёте задействованы компоненты как самой ячейки, так и соседних ячеек.

Легко подсчитать, что при объёме газа 1000 × 1000 × 1000 ячеек и примерно 100 операций с плавающей точкой для каждой ячейки, для выполнения одной итерации в секунду требуется вычислитель с производительностью около 200 GFLOPS, что намного превышает показатели современных ПЭВМ. Поэтому для решения поставленной задачи рассматривались два способа реализации: 1) на кластерах и 2) на графических платах. Каждый из них имеет свои ограничения: 1) реализация на основе дос-

тупных компьютеров, объединённых локальной сетью, уменьшает интенсивность обмена данными между вычислительными узлами; 2) реализация на графических платах ограничена объёмом памяти на самих платах и объёмом физически доступной оперативной памяти, а также задержками при копировании данных из оперативной в память графической платы и обратно.

## Общая схема GPU-алгоритма

Рассмотрим детали применения технологии GPGPU (General Programming on Graphics Processing Units) для аппаратного ускорения нейросетевых алгоритмов [19] в наших частных задачах.

Для привлечения в качестве аппаратного ускорителя графического процессора необходимо загрузить в графическую плату поочерёдно все слои трёхмерного объёма, причём для каждого слоя необходимо иметь в памяти графической платы соседний слой «слева» и соседний слой «справа» (см. рис. 2).

### Алгоритм на кластере

Для реализации управления узлами кластера и обмена между узлами использовалась технология MPI (Message Processing Interface). При организации кластерных вычислений в данном классе задач применяется пространственная декомпозиция всего объёма газа по узлам. На каждом из узлов вычисляется только некоторая часть области, в итоге возникает необходимость на каждом узле дополнительно хранить крайние слои, расчёт которых производится на соседнем узле, а после каждой итерации необходим обмен этими слоями. В остальном алгоритм полностью совпадает с алгоритмом вычислений на графических платах, но вычисления происходят на центральном процессоре (CPU).

В случае, если на узле доступно более одного CPU, поток вычислений дополнительно разбивается на потоки в соответствии с числом CPU. Также выделяются в отдельные потоки обмены с соседними узлами для максимально возможного совмещения обменов с вычислениями. Однако из-за существенных различий между подобластями газа, которые выделяются разным узлам, возникает необходимость динамической балансировки нагрузки между узлами.

### Результаты экспериментов

В результате экспериментов удалось на одной графической плате NVIDIA 7800 GTX получить ускорение в 2,5 раза по сравнению с оптимизированным алгоритмом, выполняемым на процессоре Pentium 4 (3 ГГц). На кластере из 10 узлов масштабируемость составила 90%, тогда как на двух узлах значение составило 98%. К сожалению, существующие драйверы для графических плат не позволили получить ускорение расчётов в режиме SLI (Scalable Link Interface). Поэтому была испытана реализация смешанного алгоритма распараллеливания на кластере из двух узлов, которые были оснащены графическими платами. Масштабируемость составила 90%. Это позволяет предполагать, что можно организовать параллельные вычисления на кластерной системе из таких узлов и после решения проблем с динамическим масштабированием данная конфигурация станет самой высокопроизводительной. Безусловно, существует ряд ограничений в применимости такого метода распараллеливания вычислений. Однако при этом остаётся возможность наращивания вычислительной мощности так же, как в обычной кластерной системе, без проектирования специализированных аппаратных средств.

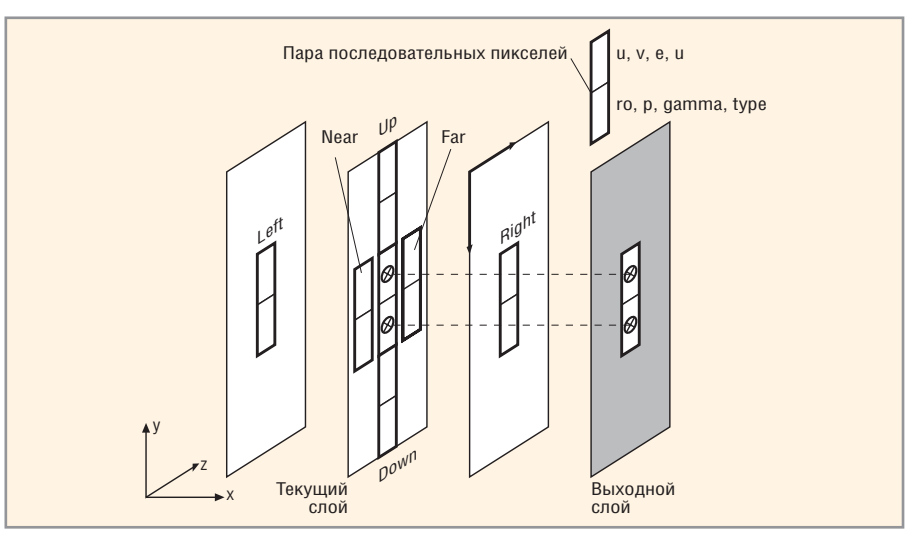


**Рис. 2. Схема представления структуры данных в текстурах графической платы**

## Заключение

В ближайшем будущем ведущие производители GPU намерены обеспечить 64-разрядную точность на операциях с плавающей точкой. Предполагается [4], что через 5-10 лет графические и центральные процессоры «сойдутся» в едином продукте. Один кристалл будет содержать в себе набор разнородных ядер, как выделенных вычислительных, так и графических, и ядер общего назначения.

Современные и последующие поколения графических процессоров являются предпочтительными для эффективной реализации на них нейрокомпьютерных конфигураций по сравнению с многоядерными микропроцессорами типа CELL и Intel Core. Это объясняется, в частности, упрощением структуры параллельных ядерных элементов GPU, что позволяет реализовать – при одинаковых техпроцессах и площадях кристаллов – большее число ядер и обеспечивать лучшие показатели производительность/стоимость. Вышесказанное можно отнести и к будущим гибридным мультипроцессорам с разнородными вычислительными и графическими ядрами.

В настоящей статье изложены результаты, полученные в рамках проекта по разработке пакета программ для решения инженерных задач с плотными системами уравнений со сверхбольшим числом неизвестных. В данном проекте использовались современные графические платы с параллельной архитектурой для построения масштабируемых кластерных нейрокомпьютерных конфигураций и реализации на них нейросетевых алгоритмов обработки сложных сигналов, изображений и других задач.

Дальнейшее развитие пакета «Нейроматематика» связано с использованием средств разработки CUDA и реализацией нейросетевых алгоритмов на основе входящей в состав CUDA библиотеки BLAS. Одновременно будут реализовываться алгоритмы распараллеливания и динамической балансировки на традиционном кластере, в том числе и на кластере из узлов, оснащённых графи-

ческими платами. Это позволит в полной мере использовать возможности масштабирования графических ускорителей нового поколения, присущей только кластерным решениям, и не расходовать дополнительные средства на создание специализированных кластерных систем.

## Литература


1. *Аляутдинов М.А., Галушкин А.И., Назаров Л.Е.* Методы распараллеливания и программно-аппаратной реализации нейросетевых алгоритмов обработки изображений. Нейрокомпьютеры. 2003. № 2. С. 3–21.
2. *Зюбин В.* Многоядерные процессоры и программирование. Открытые системы. 2005. № 07–08.
3. The PeakStream Platform. http://www.peakstreaminc.com/product/overview.
4. *Воробьев А., Медведев А.* NVIDIA GeForce 8800 GTX (G80). http://www.ixbt.com/video2/g80-part1.shtml.
5. NVIDIA CUDA Homepage. http://www.nvidia.ru/object/cuda.html.
6. *Kahle J.A., Day M.N., Hofstee H.P., Johns C.R., Maeurer T.R., and Shippy D.* Introduction to the Cell multiprocessor. IBM J. of research and development. POWER5 and Packaging. 2005. Vol. 49. No. 4/5.
7. Официальный релиз Core 2 Extreme QX6700 aka Kentsfield.: 03.11.2006. http://www.webscaner.ru/newsprint-1348.html.
8. *Романченко В.* Эволюция многоядерной процессорной архитектуры Intel Core: Conroe, Kentsfield, далее по расписанию. http://www.3dnews.ru/cpu/new_core_conroe/print.
9. *Фомин А.* Четырёхъядерные процессоры Intel Kentsfield. Intel: 32 ядра в 2010 г. http://www.3dnews.ru/news/intel_32_yadra_v_2010_godu.
10. *Озеров С.* Терапроцессоры Intel Developer Forum Fall 200. Сан-Франциско, 2006.
11. *Eichenberger A.* et al., Optimizing Compiler for the Cell Processor, PACT 2005.
12. *Аляутдинов М.А., Галушкин А.И., Троепольская Г.В.* Перспективные программно-аппаратные эмуляторы нейронных сетей на базе мультиядерных микропроцессоров с параллельной архитектурой. 9-я Международная конференция «Цифровая обработка сигналов и её применение» DSPA-2007. Москва.
13. *Аляутдинов М.А, Галушкин А.И., Троепольская Г.В.* Использование графических процессоров с параллельной архитектурой для построения масштабируемых нейрокомпьютерных конфигураций. Нейрокомпьютеры, 2006. № 8–9. С. 18–28.
14. *Пантюхин Д.В.* Использование графических ускорителей для общематематических вычислений. 9-я Международная конференция «Цифровая обработка сигналов и её применение». DSPA-2007. Москва.
15. *Скрибцов П.В., Воронков И.М.* Аппаратное ускорение алгоритмов решения уравнений газовой динамики с применением графических процессоров. 9-я Международная конференция «Цифровая обработка сигналов и её применение», Москва DSPA-2007.
16. *Воронков И.М.* Моделирование нейросетевых вычислений на кластерных системах. 9-ая Международная конференция «Цифровая обработка сигналов и её применение» DSPA-2007. Москва.
17. *Белоцерковский О.М., Давыдов Ю.М.* Метод крупных частиц в газовой динамике. Наука, 1982.
18. *Скрибцов П.В.* Аппаратное ускорение нейросетевых алгоритмов с применением графических процессоров. Труды III международной конференции «Параллельные вычисления и задачи управления». М.: Институт проблем управления РАН, 2006.