# Информационные технологии

УДК 004.032.24:004.272

## Архитектура комплекса конвейерно-параллельной обработки данных в гетерогенной вычислительной среде

**А. А. Талалаев, В. П. Фраленко**

*Федеральное государственное бюджетное учреждение науки*
*Институт программных систем им. А.К. Айламазяна Российской академии наук,*
*152021 Ярославская область, Переславский район, с. Веськово, ул. Петра I, д.4а*

Гетерогенная вычислительная среда использует различные типы вычислительных блоков. Примером такой среды является GPU-кластер, содержащий процессоры общего назначения (central processing unit, CPU) и графические процессоры специального назначения (graphics processing unit, GPU). Современные GPU уже сейчас значительно превосходят по производительности CPU и, несмотря на ограничения, накладываемые на разрабатываемые в рамках концепции GPGPU-вычислений (general-purpose graphics processing units), параллельные алгоритмы находят свое применение при решении задач, требующих интенсивных вычислений. Организация так называемого «GPU-кластера» может стать эффективным решением, обладающим приемлемым соотношением «цена/производительность» и, что самое важное, возможностью легкого наращивания производительности вычислительной системы.

Известно несколько видов параллелизма высокопроизводительных алгоритмов, актуальных и для GPU-кластеров, в том числе параллелизм задачи и параллелизм данных. В работе произведен анализ их применимости в качестве основы комплекса конвейерно-параллельной обработки данных. Исследованы варианты создания высокопроизводительных алгоритмов, предложена схема адаптации ранее разработанного программного комплекса к новым условиям. Библиотека алгоритмов GPU-вычислений в первую очередь должна обладать потокобезопасной реализацией (программный код является потокобезопасным, если он функционирует корректно при использовании нескольких параллельно запущенных вычислительных потоков). Важным и требующим внимания остается вопрос совместного использования ресурсов конкурирующими потоками. Для того, чтобы выявить влияние этого фактора на эффективность решения прикладной задачи, был поставлен эксперимент, выявляющий узкие места GPU-кластера при работе с конкурирующими потоками. Сделаны оценки порога эффективного наращивания числа вычислительных потоков, предполагающего дальнейшее ускорение счета.

**Ключевые слова:** графический процессор, вычислительный кластер, архитектура, потокобезопасность.

## Введение

Для анализа GPGPU-вычислений следует в первую очередь рассмотреть имеющиеся аппаратные и программные платформы. На сегодняшний день на этом рынке присутствуют две компании: AMD и NVIDIA, предлагающие свои решения как в бюджетном сегменте, так и в сегменте профессиональных, специализированных решений. Разрабатываемые этими компаниями программно-аппаратные платформы с точки зрения разработчика прикладного программного обеспечения во многом схожи, однако конкурентная борьба породила создание двух различных прикладных интерфейсов (application programming interface, API), поддержка которых требует значительно больше времени и ресурсов. Компания NVIDIA

Работа выполнена в рамках Программы фундаментальных исследований Президиума РАН №18 «Алгоритмы и математическое обеспечение для вычислительных систем сверхвысокой производительности» (проект «Исследование и разработка инструментальных программных средств мониторинга и обработки потоков данных подсистем космических аппаратов с применением суперкомпьютерных систем, оснащенных многоядерными и графическими процессорами») и при поддержке Российского фонда фундаментальных исследований (проект №12-01-31500-мол_а «Разработка инструментальных программных средств для проектирования нейросетевых прикладных систем»).

продвигает платформу CUDA (Compute Unified Device Architecture), предоставляющую разработчику низкоуровневые API для реализации алгоритмов на GPU. Ее аналогом является программная платформа Accelerated Parallel Processing (APP) компании AMD. Наиболее значимой попыткой объединения этих разнородных платформ является развивающийся стандарт OpenCL [1], который в текущих условиях выглядит перспективнее, поскольку его применение приводит к унификации разрабатываемых алгоритмов, тогда как разнообразные программные библиотеки (OpenCV, MAGMA, CULA, ArrayFire, NPP и др.), использующие GPU для ускорения счета, в большинстве случаев остаются ориентированными на конкретного производителя.

Использующие GPU программные библиотеки обеспечивают значительное преимущество при вычислениях за счет распараллеливания задач в случае возможности их декомпозиции на низком уровне. Одновременное использование набора GPU достаточно просто реализуется на системах с SMP-архитектурой, но может быть неэффективно на системах кластерного типа. Известны следующие виды параллелизма высокопроизводительных алгоритмов, которые являются актуальными и для GPU-кластеров:

1) параллелизм задачи — одновременное использование нескольких GPU, установленных на различных узлах кластерного вычислительного устройства (КВУ), для решения единой прикладной задачи;
2) параллелизм данных — использование множества GPU для конвейерной обработки потока данных.

В первом случае накладные расходы на передачу данных между узлами КВУ, имеющими значительно большую задержку чем обмен данными типа ОЗУ-GPU, могут нивелировать положительный эффект от применения GPU. Подход применим к достаточно узкому кругу решаемых задач с большими гранулами параллелизма и имеет несомненное преимущество, если исходная задача обладает высокой вычислительной сложностью, поскольку можно ожидать значительного сокращения времени счета. Во втором случае использование множества вычислительных узлов КВУ с установленными графическими ускорителями дает возможность построить эффективный конвейер обработки данных с декомпозицией задачи на более высоком уровне. Этот, достаточно универсальный вариант построения программной системы, выбран авторами в качестве основы разрабатываемого комплекса конвейерно-параллельной обработки данных.

Концепция GPGPU-вычислений нашла применение в таких сферах как математическое моделирование процессов, визуализация и анализ графических данных [2]. Эффективным, на наш взгляд, является использование ресурсов гетерогенной среды GPU-кластера, когда интенсивные вычисления используют ресурсы GPU, а остальные расчеты ведутся на CPU — возможно в параллельном режиме с применением многоядерных процессоров или многопроцессорных решений в пределах каждого вычислительного узла.

## 1. Архитектура комплекса

Ранее предложенная архитектура программно-инструментального комплекса «НСКиД» [3–5] может быть легко адаптирована к использованию ресурсов гетерогенной среды GPU-кластера (рис. 1).

Модули обработки информации, применяемые для решения прикладных задач, реализуются в виде динамических библиотек и могут использовать ресурсы GPU для проведения необходимых расчетов. Система диспетчеризации нагрузки, реализованная в ядре программной системы, поддерживает выполнение задачи в многопоточном окружений на КВУ, обеспечивая конвейерно-параллельный режим обработки данных.

Для использования GPU в рамках разработанного программного комплекса достаточным является выполнение единственного требования — библиотека алгоритмов GPU-вычислений должна обладать потокобезопасной реализацией. Это 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 организована в виде «очереди заданий», что обеспечивает большую отказоустойчивость системы, синхронизируя доступ к разделяемому ресурсу (GPU) из многопоточной среды.

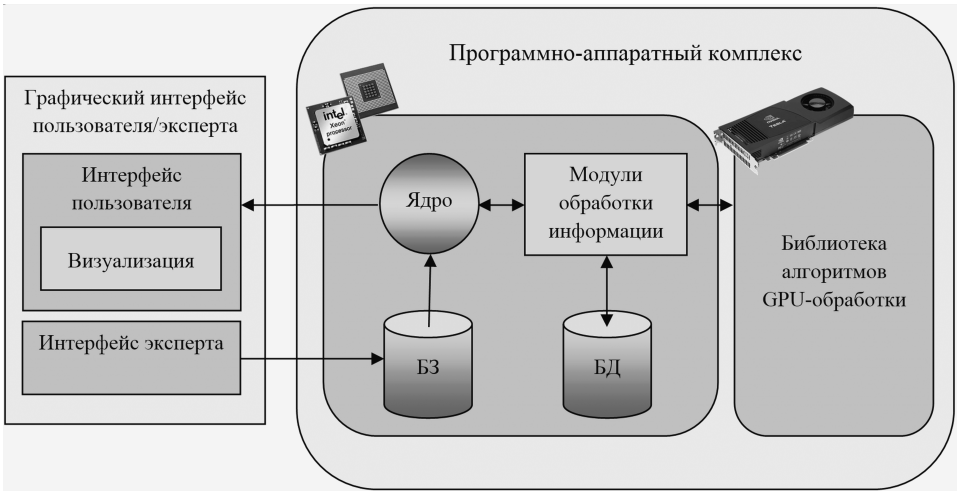


**Рис. 1. Общая архитектура программного комплекса**

## 2. Эксперименты с конкурирующими потоками

Важным и требующим внимания остается вопрос совместного использования ресурсов конкурирующими потоками. Для того, чтобы оценить влияние этого фактора на эффективность решения прикладной задачи, был поставлен следующий эксперимент. На рабочей станции, построенной на основе двухпроцессорной конфигурации Intel Xeon X5570 @ 2.93GHz и снабженной графическим ускорителем Nvidia Tesla C1060, проведено сравнительное тестирование, в котором решалась задача перемножения матриц 2048x2048 элементов. Операция выполнялась для матриц с действительными и комплексными числами одинарной и двойной точности с использованием библиотек ATLAS (при вычислениях на CPU) и CUBLAS (при вычислениях на GPU). Для увеличения нагрузки операция выполнялась многократно. Полученные в ходе тестирования результаты представлены в табл. 1–4.

**Таблица 1**

**Время счета на GPU, многопоточный режим (от 1 до 16 потоков)**

| Формат данных, сек. | Конкурирующие потоки | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 |
| Одинарная точность | 15.45 | 18.40 | 21.56 | 35.87 | 55.65 |
| Двойная точность | 20.48 | 28.42 | 41.55 | 73.05 | 127.43 |
| Комплексные числа с одинарной точностью | 22.37 | 28.68 | 37.93 | 65.56 | 107.42 |
| Комплексные числа с двойной точностью | 40.12 | 62.75 | 104.34 | 190.73 | 343.94 |

Проведенный эксперимент показал, что одновременное использование ресурсов GPU из нескольких потоков в ходе решения задачи позволяет получить определенное преимущество, однако масштабирование скорости вычислений незначительно по сравнению с CPU-реализацией. При этом порог эффективного наращивания числа вычислительных потоков, предполагающего дальнейшее ускорение

**Таблица 2**

**Ускорение счета на GPU (сравнение с последовательной реализацией)**

| Формат данных, раз | Конкурирующие потоки | | | |
|---|---|---|---|---|
| | 2 | 4 | 8 | 16 |
| Одинарная точность | 1.679 | 2.866 | 3.446 | 4.442 |
| Двойная точность | 1.441 | 1.972 | 2.243 | 2.571 |
| Комплексные числа с одинарной точностью | 1.560 | 2.359 | 2.730 | 3.332 |
| Комплексные числа с двойной точностью | 1.279 | 1.538 | 1.683 | 1.866 |

**Таблица 3**

**Время счета на CPU, многопоточный режим (от 1 до 16 потоков)**

| Формат данных, сек. | Конкурирующие потоки | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 |
| Одинарная точность | 21.01 | 21.23 | 21.49 | 21.72 | 49.36 |
| Двойная точность | 35.39 | 35.48 | 36.19 | 37.42 | 75.50 |
| Комплексные числа с одинарной точностью | 77.02 | 77.29 | 79.50 | 82.54 | 158.40 |
| Комплексные числа с двойной точностью | 141.90 | 144.97 | 145.31 | 145.58 | 279.53 |

**Таблица 4**

**Ускорение счета на CPU (сравнение с последовательной реализацией)**

| Формат данных, раз | Конкурирующие потоки | | | |
|---|---|---|---|---|
| | 2 | 4 | 8 | 16 |
| Одинарная точность | 1.979 | 3.911 | 7.738 | 6.810 |
| Двойная точность | 1.995 | 3.912 | 7.566 | 7.500 |
| Комплексные числа с одинарной точностью | 1.993 | 3.875 | 7.465 | 7.780 |
| Комплексные числа с двойной точностью | 1.958 | 3.906 | 7.798 | 8.122 |

счета, для GPU выше. Для CPU уже при 16 конкурирующих потоках наблюдается снижение эффективности на операциях с действительными числами.

## Заключение

GPU-кластер является достаточно эффективным решением, обладающим приемлемым соотношением «цена/производительность», и позволяет легко наращивать производительность системы. Среди рассмотренных в работе вариантов реализации высокопроизводительных алгоритмов на основе GPU-кластеров выбрана модель параллелизма данных, позволяющая строить эффективный конвейер обработки, поддерживаемый программным комплексом. Предложенная архитектура позволяет полноценно использовать ресурсы гетерогенной среды GPU-кластера. Проведенный эксперимент выявил характерные для рассмотренной архитектуры особенности, которые целесообразно учитывать при решении прикладных задач.

## Литература

1. OpenCL official site. — `www.khronos.org/opencl`.
2. GPU Applications. — `http://www.nvidia.com/object/gpu-applications.html?All`.

3. Свидетельство о государственной регистрации программы для ЭВМ №2012613261. — Нейросетевая система контроля телеметрической информации, диагностики подсистем космических аппаратов, обработки космических снимков (ПС НСКиД). Нейросетевая система контроля телеметрической информации, диагностики подсистем космических аппаратов, обработки космических снимков (ПС НСКиД). [Svideteljstvo o gosudarstvennoyj registracii programmih dlya EhVM No2012613261. — Neyjrosetevaya sistema kontrolya telemetricheskoyj informacii, diagnostiki podsistem kosmicheskikh apparatov, obrabotki kosmicheskikh snimkov (PS NSKiD). Neyjrosetevaya sistema kontrolya telemetricheskoyj informacii, diagnostiki podsistem kosmicheskikh apparatov, obrabotki kosmicheskikh snimkov (PS NSKiD). ]
4. *Талалаев А. А.* Организация конвейерно-параллельных вычислений для обработки потоков данных // Информационные технологии и вычислительные системы. — 2011. — № 1. — С. 8–13. [*Talalaev A. A.* Organizaciya konveyjerno-paralleljnihkh vihchisleniyj dlya obrabotki potokov dannihkh // Informacionnihe tekhnologii i vihchisliteljnihe sistemih. — 2011. — No 1. — S. 8–13. ]
5. *Хачумов В. М., Фраленко В. П.* Высокопроизводительная обработка изображений на кластерных устройствах // Нейрокомпьютеры: разработка и применение. — 2012. — № 6. — С. 38–45. [*Khachumov V. M., Fralenko V. P.* Vihsokoproizvoditeljnaya obrabotka izobrazheniyj na klasternihkh ustroyjstvakh // Neyjrokompjyuterih: razrabotka i primenenie. — 2012. — No 6. — S. 38–45. ]

UDC 004.032.24:004.272

# The Architecture of a Parallel-Pipeline Data Processing Complex for Heterogeneous Computing Environment

**A. A. Talalaev, V. P. Fralenko**

*Institute of program systems of the Russian Academy of Science,*
*152021 Pereslavl-Zalessky Yaroslavl Region, Veskovo, Peter I, 4a*

A heterogeneous computing environment uses various types of computational units. An example of such environment is a GPU-cluster that contains general-purpose processors (central processing unit, CPU) and graphics processing units for special purposes (GPU). Today's GPU is already far superior CPU performance and, despite the limitations imposed by developed under the concept of GPGPU-computing (general-purpose graphics processing units), parallel algorithms find their application in solving problems that require intensive computation. Organization of the so-called "GPU-cluster" may be an effective solution that have an acceptable "price/performance" ratio and, that most importantly, an ability to easily scale a computer system performance.

There are several types of high-performance algorithms for concurrency that relevant for GPU-cluster too (including a task and data parallelism). In this paper produced an analysis of their applicability as a basis set of parallel-pipeline computations data processing. Investigated a variants of high-performance algorithms building, proposed previously developed software adaptation scheme for a new conditions. Library of GPU-computing algorithms in the first place should have a thread-safe implementation (the code is thread-safe if it functions work correctly with multiple running parallel computing threads). An important and needs attention is the question of competing threads resource sharing. In order to assess the impact of this factor on the effectiveness of applied problem, we performed an experiment, identifying GPU-cluster competing threads dealing bottlenecks. Have been estimated the effective threshold for increasing the number of processing threads that is expected to a further calculations accelerating.

**Key words and phrases:** graphics processing unit, compute cluster, architecture, thread safety.