ОБЗОР И АНАЛИЗ ПРОГРАММНЫХ ПРОДУКТОВ ДЛЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Гоголев Е.С.

Гоголев Евгений Сергеевич – студент магистратуры,
факультет прикладной математики и информатики,
Московский авиационный институт,
УЦ «Интеграция», г. Серпухов

Аннотация: *в статье рассмотрены и проанализированы два программных продукта, выявлены их достоинства и недостатки. Это программы CUDA и ROOT/PRooF.*
Ключевые слова: *параллельные вычисления, параллельное программирование, анализ.*

CUDA (Compute Unified Device Architecture) − программно-аппаратная архитектура параллельных вычислений, благодаря которой можно увеличить вычислительную производительность из-за применения графических процессоров фирмы NVIDIA [1].

CUDA SDK позволяет программистам реализовывать на специальном упрощённом диалекте языка программирования Си алгоритмы, выполнимые на графических процессорах NVIDIA, и включать специальные функции в текст программы на Си. Архитектура CUDA даёт разработчику возможность по своему усмотрению организовывать доступ к набору инструкций графического ускорителя и управлять его памятью.

В архитектуре CUDA используется модель памяти грид, кластерное моделирование потоков и SIMD-инструкции. Применима не только для высокопроизводительных графических вычислений, но и для различных научных вычислений с использованием видеокарт nVidia. Ученые и исследователи широко используют CUDA в различных областях, включая астрофизику, вычислительную биологию и химию, моделирование динамики жидкостей, электромагнитных взаимодействий, компьютерную томографию, сейсмический анализ и многое другое. В CUDA имеется возможность подключения к приложениям, использующим OpenGL и Direct3D. CUDA − кроссплатформенное программное обеспечение для таких операционных систем как Linux, Mac OS X и Windows.

После обзора и анализа данного продукта были выявлены как достоинства программы, так и недостатки.

Итак, преимущества данной программы:

− Интерфейс основан на стандартном языке программирования Си, что упрощает процесс изучения архитектуры.
− Разделяемая между потоками память (shared memory) размером в 16 Кб может быть использована под организованный пользователем кэш с более широкой полосой пропускания, чем при выборке из обычных текстур.
− Более эффективные транзакции между памятью центрального процессора и видеопамятью.
− Полная аппаратная поддержка целочисленных и побитовых операций.
− Поддержка компиляции GPU кода средствами открытого LLVM.

Недостатки были обнаружены следующие:

− Сложность программирования для CUDA.
− Привязка именно к картам NVidia.
− Вложенный механизм распараллеливания функций и запуска потоков, особенности работы с памятью, конфигурацию оборудования.
− Bottlenecked скорости в шине между CPU и GPU.
− Скептическое отношение сообщество по отношению к CUDA.
− Малое количество разработчиков.
− Функции не поддерживают рекурсии.

Другое программное средство для параллельных вычислений, которое было рассмотрено это ROOT/PRooF.

ROOT представляет собой объектно-ориентированную среду для анализа и визуализации данных. В её состав входит множество компонентов и инструментов, позволяющих:

− построение гистограмм, графиков и функций;
− “подгонку” теоретических кривых под экспериментальные данные и минимизацию функций;
− обработку изображений;
− доступ к базам данных;
− использование нейронных сетей;
− хранение и обработка больших объемов данных;
− выполнять математические вычисления благодаря встроенным математическим библиотекам;

- параллельные вычисления благодаря PROOF;
- интегрировать с Mathematica, Ruby, Python.

Также хочется отметить, что для написания скриптов используется язык программирования C++. Это позволяет пользователям, знакомым с этим языком, не тратить время на изучение специального языка, а сразу приступить к работе с ROOT. Включение в пакет интерпретатора C++ CINT значительно увеличило гибкость пакета, так как позволило использовать средства ROOT в интерактивном режиме или посредством написания скриптов.

Немаловажным фактором является бесплатность и кроссплатформенность ROOT, а также наличие посторонних библиотек. Впрочем, и саму ROOT можно использовать как библиотеку для написания своих программ.

Хочется отметить постоянную и полную поддержку разработчиками. ROOT постоянно обновляется. На сайте можно найти полное руководство пользователя, множество различных примеров и даже скачать исходные коды системы.

PROOF-кластер строится по стандартной схеме master-slave. Благодаря многоуровневой архитектуре, позволяющей создать иерархию из master и submaster узлов, такой подход может быть легко адаптирован к широкому диапазону виртуальных кластеров, географически распределённых между доменами и гетерогенными машинами (GRID). Клиентом системы является пользователь, который хочет использовать ресурсы сайта, чтобы выполнить свою задачу [2, 67]. Master – это точка входа к вычислительному средству: он разбирает запросы клиента, распределяет работу между ресурсами, собирает и соединяет результаты. Координация работы отдельных серверов достигается за счет использования специального протокола передачи данных PROOF.

Пользователь, работая в сессии программы ROOT, может запускать процессы, которые связываются с PROOF-кластером и подают запросы на обработку заданий. Получив запрос на обработку задания, на мастере и на рабочих узлах для каждой сессии пользователя стартует специальное ROOT-приложение – proofserv [3, 485]. Процесс, исполняющийся на мастере, координирует работу между рабочими узлами и объединяет результаты в единое целое. На рабочих узлах процесс proofserv делает непосредственно вычислительную работу, обрабатывая отдельные задания.

Список литературы

1. *Давлеткалиев Рахим*. Введение в параллельные вычисления, 2011. [Электронный ресурс]. Режим доступа: http://habrahabr.ru/post/126930/ (дата обращения: 26.04.2017).
2. *Tanenbaum Andrew S., Van Steen Maarten.* "Destributed systems. Principles and paradigms". Санкт-Петербург: Питер, 2003. 877 с. (Классика computer science). ISBN 5-272-00053-6.
3. *Amdahl G.* Validity of the Single Processor Approach to achieving Large-Scale Computing Capabilities. In: AFIPS Conference Proceedings. Vol. 30. Pp. 483-485 (1967).