А. О. Молоталиев, В. А. Коковин, А. Н. Сытин

ПЕРСПЕКТИВА ВНЕДРЕНИЯ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ ОБЩЕГО НАЗНАЧЕНИЯ В ПРИЛОЖЕНИЯ СРЕДСТВ АВТОМАТИЗАЦИИ

Филиал «Протвино» университета «Дубна»
Кафедра автоматизации технологических процессов и производств

В статье рассмотрены особенности использования графических процессоров общего назначения в системах автоматизации , показаны преимущества этого использования за счет большого количества аппаратных ядер и параллельного выполнения программных инструкций.

Введение

Видеочипы в параллельных математических расчётах пытались использовать довольно давно. Самые первые попытки такого применения были крайне примитивными и ограничивались использованием некоторых аппаратных функций, таких, как растеризация (получение растровых изображений) и *Z*-буферизация (способ представления объемности). Но в настоящее время, с появлением шейдеров (программ представления двух-мерной графики объемной), появилась необходимость ускорять вычисления матриц. В 2003 г. на *SIGGRAPH* [1] под вычисления на *GPU (Graphics Processing Unit)* была выделена отдельная секция, и она получила название *GPGPU* (*General-Purpose Computation on GPU*) — универсальные вычисления на *GPU*).

Наиболее известен *BrookGPU* [2] — компилятор потокового языка программирования *Brook*, созданный для выполнения неграфических вычислений на *GPU*. До его появления разработчики, использующие возможности видеочипов для вычислений, выбирали один из двух распространённых *API*: *Direct3D* или *OpenGL*. Это серьёзно ограничивало применение *GPU*, ведь в *3D*-графике используются шейдеры и текстуры, о которых специалисты по параллельному программированию знать не обязаны, они используют потоки и ядра. Компилятор *Brook* смог помочь в решении этих задач. Эти потоковые расширения к языку *C*, разработанные в Стэндфордском университете, скрывали от программистов трёхмерный *API* и представляли видеочип в виде параллельного сопроцессора. Компилятор обрабатывал файл *.br* с кодом *C++* и расширениями, производя код, привязанный к библиотеке с поддержкой *DirectX*, *OpenGL* или *x86*.

Естественно, у *Brook* было множество недостатков, о которых будет сказано ниже. Но даже просто появление этого компилятора вызвало пристальное внимание фирмы *NVIDIA* [3] к инициативе вычислений на *GPU*. Развитие этих возможностей в дальнейшем серьёзно изменило рынок, открыв целый новый его сектор — параллельные вычислители на основе видеочипов.

Позже некоторые исследователи из проекта *Brook* влились в команду разработчиков *NVIDIA*, чтобы представить программно-аппаратную стратегию параллельных вычислений, открыв новый сектор рынка. И главным преимуществом этой инициативы *NVIDIA* стало то, что разработчики детально знают все возможности *GPU*, и в использовании графического *API* нет необходимости, а работать с аппаратным обеспечением можно напрямую — при помощи драйвера. Результатом усилий этой команды стала *NVIDIA CUDA* (*Compute Unified Device Architecture*) [4]. Стоит отметить, что помимо *CUDA* существуют и другие технологии для массивно-параллельных вычислений. (Примером этих технологий может быть *OpenCL* и *DirectCompute*.

В настоящее время большинство систем числового программного управления на базе индустриальных компьютеров (*ICNC — **Industrial** Computer Numerical Control [5]*) достигли предела своего развития, при этом начинает не хватать вычислительных возможностей центрального процессора (ЦП). Например, при увеличении числа интерполирующих осей, нагрузка на центральный процессор увеличивается экспоненциально.

Преимущества *GPU* перед центральными процессорами

Рост частот универсальных процессоров упёрся в физические ограничения и высокое энергопотребление. Таким образом, увеличение их производительности всё чаще происходит за счёт размещения нескольких ядер в одном чипе. Продаваемые сейчас процессоры содержат до шестнадцати ядер, используют *MIMD* (*Multiple Instruction stream, Multiple Data stream*) — множественный поток команд и данных: предназначены для обычных приложений. Каждое ядро работает отдельно от остальных, исполняя разные инструкции для разных процессов.

Специализированные векторные возможности (*SSE2* и *SSE3*) для четырехкомпонентных (одинарная точность вычислений с плавающей точкой) и двухкомпонентных (двойная точность) векторов появились, из-за возросших требований графических приложений, в первую очередь, в универсальных процессорах. Именно поэтому для определённых задач выгоднее применение *GPU*, ведь они изначально сделаны для них.

Например, в видеочипах *NVIDIA* основной блок — это мультипроцессор с восемью-десятью ядрами и сотнями *ALU* в целом, несколькими тысячами регистров и небольшим количеством разделяемой общей памяти. Кроме того, видеокарта содержит быструю глобальную память с доступом к ней всех мультипроцессоров, локальную память в каждом мультипроцессоре, а также специальную память для констант.

Самое главное — эти несколько ядер мультипроцессора в *GPU* являются *SIMD* — ядрами (одиночный поток команд, множество потоков данных). И эти ядра исполняют одни и те же инструкции одновременно. Такой стиль программирования является обычным для графических алгоритмов и многих научных задач, но требует специфического программирования. Зато такой подход позволяет увеличить количество исполнительных блоков за счёт их упрощения.

Архитектурные отличия *GPGPU ICNC* от классической *CNC*

На данный момент большинство программно-реализованных функций в системах ЧПУ (рис. 1) выполняются центральным процессором.

В новой модели выполнения часть наиболее ресурсоемких задач — ядро ЧПУ, программно-реализованный контролер автоматики — перенесены на видеопроцессор. Достоинством данного решения состоит в том, что оно позволяет организовать выполнение нескольких виртуальных ЧПУ-систем для обслуживания станков разной номенклатуры, но имеющих общий участок обработки заготовки детали. Фактически внедрения массивно-параллельных вычислителей открывает возможность упрощения системы управления и инфраструктуры за счет дифференцирования функций вычислителей от коммуникационных операций.

Экспериментальные данные исследования

В результате проведенного эксперимента было выявлено заметное ускорение выполнения задач, связанных с вычислением больших объемов данных, при этом производительность обработки выросла в среднем более чем в 30 раз.

Наибольший выигрыш заметен в сплайн-интерполяции и составляет 120-кратное увеличение скорости выполнения по сравнению с однопроцессорной системой. Также стоит отметить сокращение объема кода вычислителя, по сравнению с классической реализацией.

Пример кода для центрального процессора

```
for (int i = 0; i < Nummelemets; i++){
Out[i] = LinearBezier(PX[0],PX[1],i*eps);
  }
```

Пример кода для графического процессора

```
__global_  Bezier1stOrder(int*  output_vector,const  int P0,const int P1,float T)
{
int gid = threaded.x + (blockdim.x * blockid.x));
output_vector[tid] = (1-(T*tid))* P0) + P1*(T*tid)
                                                    }
```

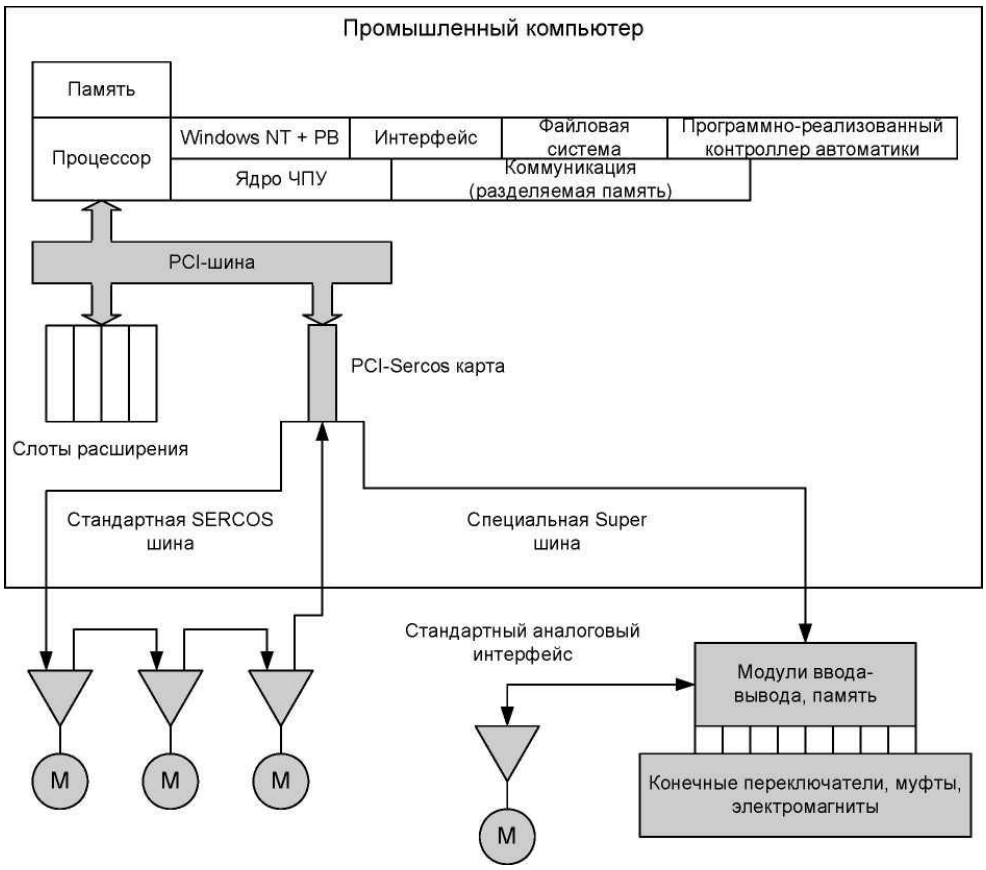

Рис. 1 Модель выполнения программ на основе ЦП

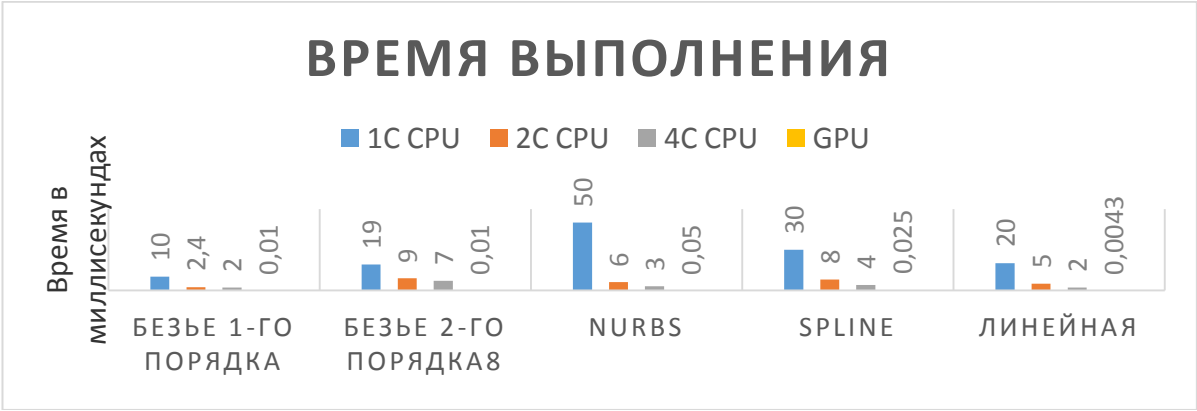

Рис. 2 Результаты экспериментов

На рис. 2 представлены результаты проведённого исследования (вычисление различных функций разными вычислителями), которые показывают преимущество использования массивно-параллельных вычислений в системах числового программного управления.

Библиографический список

1. http://www.siggraph.org/s2003/
2. http://graphics.stanford.edu/projects/brookgpu/
3. www.nvidia.ru/
4. http://www.nvidia.ru/object/cuda-parallel-computing-ru.html
5. http://www.ccas.ru/paral/mimd/mimd.html