Contribution ID: **248** Type: **not specified**

Высокопроизводительные вычисления на математических сопроцессорах и графических ускорителях с использованием Python

Monday, 3 July 2023 17:15 (15 minutes)

УДК 519.6+004.42

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА МАТЕМАТИЧЕСКИХ СОПРОЦЕССОРАХ И ГРАФИЧЕСКИХ УСКОРИТЕЛЯХ С ИСПОЛЬЗОВАНИЕМ PYTHON

С. В. Борзунов, А. В. Романов, С. Д. Кургалин, К. О. Петрищев

ФГБОУ ВО «Воронежский государственный университет»

В настоящей работе рассмотрено применение модулей языка программирования Python для решения ресурсоемких задач. На примере перемножения вещественных матриц продемонстрированы методы высокопроизводительных вычислений на базе математических сопроцессоров и графических ускорителей. Показано, что существует удобный интерфейс организации вычислений путем вызова исполняемого кода из скрипта на языке Python. Тем не менее, использование полного функционала графических ускорителей труднее осуществить указанным образом, что объясняется особенностями работы потоковых ядер. Использование алгоритмического языка Python в высокопроизводительных вычислениях во многих случаях значительно расширяет удобство создания, отладки и тестирования программного кода.
Ключевые слова: высокопроизводительные вычисления, суперкомпьютер, компьютерный кластер, язык программирования Python, CUDA

Введение

Многие современные вычислительные комплексы, входящие в список Top500 ведущих суперкомпьютеров мира, собраны по гибридной схеме [1, 2]. Они имеют в своем составе не только универсальные процессоры, но и энергоэффективные математические сопроцессоры, такие как Intel Xeon Phi или Nvidia Tesla. Подобные машины представляют собой пример устройств, созданных специально для выполнения массивных параллельных вычислений. Например, сопроцессор Intel Xeon Phi предоставляет возможность производить вычисления с использованием до 240 логических ядер, а математический сопроцессор Nvidia Tesla A100 включает в себя 6912 потоковых ядер CUDA. Такие суперкомпьютеры имеют достаточно сложную архитектуру взаимодействия между процессором и сопроцессором, которая сильно усложняет процесс подготовки и поддержки программного кода. Программирование подобных систем с использованием полного функционала средств распараллеливания вычислений имеет свои особенности, поскольку современные GPU, в отличие от центральных процессоров, представляют собой массивно-параллельные вычислительные устройства с относительно большим количеством вычислительных ядер и иерархически организованной собственной памятью.

Суперкомпьютерный центр Воронежского государственного университета (ВГУ), созданный в 2002 году [3, 4], имеет в своем составе высокопроизводительный вычислительный кластер, состоящий из 10 узлов, в каждом из которых по два 12-ядерных процессора, 128 Гбайт оперативной памяти и SSD-диск 256 Гбайт. При этом семь узлов кластера содержат по 2 ускорителя Intel Xeon Phi, а три остальных узла –по 2 ускорителя Nvidia Tesla. Узлы кластера объединены сетью InfiniBand. Кластер используется как для проведения научных вычислений, так и в учебном процессе факультета компьютерных наук ВГУ.

В связи с тем, что в узлах кластера находятся ускорители указанных двух типов, актуальной является задача обеспечить их эффективное использование.

Методы высокопроизводительных вычислений
с использование модулей Python

Следуя современным тенденциям разработки программных инструментов более высокого уровня для упрощения программирования сложных вычислительных систем, созданы специальные модули языка Python, которые позволяют упростить методы работы с сопроцессором. В качестве примеров таких расширений укажем на модули PyMIC и PyCUDA [5]. Принцип действия этих модулей совпадает, он заключается в предоставлении интерфейса к основным операциям процессор/сопроцессор.
Разумеется, реализация перемножения вещественных матриц на языке Python не может похвалиться высокой скоростью вычислений, поэтому математические операции подобного рода выполняются, как правило, средствами сторонних библиотек, чаще всего, написанных на языках C или Fortran [6]. Примером могут служить вычисления, производимые с помощью пакета NumPy. В этом случае в качестве сторонних библиотек используются подпрограммы, реализованные на языке Fortran. Соответственно, первым шагом к переносу вычислений на сопроцессор будет реализация пользовательской библиотеки на компилируемом языке и ее интеграция в код на языке Python.
Наиболее доступными с точки зрения простоты программирования являются вычисления с использованием математического сопроцессора Intel Xeon Phi. Это объясняется сходством архитектуры сопроцессора с универсальными процессорами фирм Intel и AMD, что позволяет использовать уже известные приемы программирования.
Рассмотрим работу с таким сопроцессором на примере задачи перемножения двух вещественных матриц.
В листинге 1 представлен вариант исходного кода пользовательской библиотеки для интеграции с модулем PyMIC. Заметим, что в этом случае код практически не отличается от стандартной реализации перемножения матриц на языке C.
Листинг 1

include <pymic$_k ernel.h >$

include <stdio.h>

include <complex.h>

PYMIC_KERNEL

void multiplication (const double *A, const double*B, double *C, const int*nrows, const int *ncols){

```
for(int i = 0; i < *nrows; i++)
    for(int j = 0; j < *ncols; j++)
    \{
        for(int k = 0; k < *ncols; k++)
        C[i][j] += A[i][k] * B[k][j];
    \}

   \}
```

Библиотека выполняет расчеты на сопроцессоре Intel Xeon Phi, получая данные из программы на языке Python, и возвращает обратно результат вычисления.
Код программы с вызовом представленной ранее библиотеки (libmult.so) и генерацией матриц средствами пакета NumPy представлен в листинге 2.
Листинг 2

```
import pymic as mic
import numpy as np

device = mic.devices[0]
library = device.load_library("libmult.so")

stream = device.get_default_stream()

nrows = 1024
ncols = 1024

a = np.random.random(size = (nrows,ncols))
b = np.random.random(size = (nrows,ncols))
c = np.zeros(size = (nrows, ncols))
```

```
offl_a = stream.bind(a)
offl_b = stream.bind(b)
offl_c = stream.bind(c)

offl_c.update_device()

stream.invoke(library.multiplication, offl_a, offl_b, offl_c, nrows,ncols)
stream.sync()

offl_c.update_host()
stream.sync()
```

Учитывая возможность использования для вычислений до 240 потоков (одно физическое ядро зарезервировано под нужды операционной системы), сопроцессор Intel Xeon Phi предоставляет широкие возможности в исследовании вопросов параллельных вычислений без необходимости глубокого понимания его архитектуры и специальных диалектов языка C.
Подготовка программного кода, ориентированного на работу с GPU фирмы Nvidia, значительно сложнее. Основная трудность написания кода для сопроцессора Tesla объясняется особенностями работы потоковых ядер CUDA.
В листинге 3 приведен текст программы перемножения двух матриц, написанный для сопроцессора Nvidia Tesla с применением модуля PyCUDA.
Листинг 3

```
import pycuda.driver as cuda
import pycuda.autoinit
from pycuda.compiler import SourceModule
import numpy

(n, m, p) = (3, 4, 5)

n = numpy.int32(n)
m = numpy.int32(m)
p = numpy.int32(p)

a = numpy.random.randint(2, size=(n, m))
b = numpy.random.randint(2, size=(m, p))
c = numpy.zeros((n, p), dtype=numpy.float32)

a = a.astype(numpy.float32)
b = b.astype(numpy.float32)
a_gpu = cuda.mem_alloc(a.size * a.dtype.itemsize)
b_gpu = cuda.mem_alloc(b.size * b.dtype.itemsize)
c_gpu = cuda.mem_alloc(c.size * c.dtype.itemsize)

cuda.memcpy_htod(a_gpu, a)
cuda.memcpy_htod(b_gpu, b)
mod = SourceModule("""
__global__ void multiply( int n, int m, int p, float *a, float*b, float *c )
{
int idx = p*threadIdx.x + threadIdx.y;

c[idx] = 0.0;
    for(int k=0; k<m; k++)
       c[idx] += a[m*threadIdx.x+k]
                 *b[threadIdx.y+k*p];
\}
```""")

```
func = mod.get_function("multiply")
func(n, m, p, a_gpu, b_gpu, c_gpu, block=(int(n), int(p), 1), \
grid=(1, 1), shared=0)

cuda.memcpy_dtoh(c, c_gpu)
```

Как и в случае кода для Intel Xeon Phi, программа состоит из инициализации данных, их пересылки на сопроцессор и обратно, а также из ядра, которое осуществляет вычисления. Однако, математическое ядро здесь сильно отличается от аналога для сопроцессора Intel из-за разницы архитектур.

Заключение

Использование алгоритмического языка Python в высокопроизводительных вычислениях во многих случаях значительно расширяет удобство создания, отладки и тестирования программного кода. Более того, для обучения специалистов работе на современных суперкомпьютерах целесообразно начинать обучение с более простых в программировании сопроцессоров фирмы Intel с переходом на сопроцессоры

Nvidia, ставшими сегодня признанным стандартом в отрасли. В итоге, использование для операций верхнего уровня модулей PyMIC и PyCUDA позволяет существенно упростить работу с суперкомпьютером, предоставить бесшовную интеграцию с инструментами популярного языка программирования Python.
Литература

1. Korenkov, V. The JINR distributed computing environment / V. Korenkov, A. Dolbilov, V. Mitsyn [et al.]
2. . . Using the resources of the Supercomputer Center of Voronezh State University in learning pro
3. . . / . . //
4. . . - / . . // Distributed omputing and GRID-Te
5. Pycuda 2022.2.2 documentation // 2022. – URL: https://documen.tician.de/pycuda/ (: 12.05.2023)
6. , . . : / . . , . . . – - : - , 2019. – 256 .

Сведения об авторах

Борзунов Сергей Викторович, к.ф.-м.н., доц. кафедры цифровых технологий факультета компьютерных наук Воронежского государственного университета. 394018, Воронеж, Университетская пл., 1.
E-mail: sborzunov@gmail.com. Тел.: 8 (473) 220-83-84.

Романов Александр Викторович, ст. преп. кафедры цифровых технологий факультета компьютерных наук, вед. инженер суперкомпьютерного центра Воронежского государственного университета. 394018, Воронеж, Университетская пл., 1.
E-mail: alphard.rm@gmail.com. Тел.: 8 (473) 220-83-84.

Кургалин Сергей Дмитриевич, д.ф.-м.н., проф., зав. кафедрой цифровых технологий факультета компьютерных наук Воронежского государственного университета. 394018, Воронеж, Университетская пл., 1.
E-mail: kurgalin@bk.ru. Тел.: 8 (473) 220-83-84.

Петрищев Константин Олегович, студ. факультета компьютерных наук Воронежского государственного университета, г. Воронеж. 394018, Воронеж, Университетская пл., 1.
E-mail: vrn.kostyan.p@mail.ru. Тел.: 8 (961) 616-97-93.

HIGH-PERFORMANCE COMPUTING ON MATHEMATICAL CO-PROCESSORS AND GRAPHIC ACCELERATORS USING PYTHON

S. V. Borzunov, A. V. Romanov, S. D. Kurgalin, K. O. Petrishchev

Voronezh State University

The paper considers the use of Python programming language modules for solving resource-intensive tasks. (

Keywords: high performance computing, supercomputer, computer cluster, Python programming language, CUDA

Summary

В настоящей работе рассмотрено применение модулей языка программирования Python для решения ресурсоемких задач. На примере перемножения вещественных матриц продемонстрированы методы высокопроизводительных вычислений на базе математических сопроцессоров и графических ускорителей. Показано, что существует удобный интерфейс организации вычислений путем вызова исполняемого кода из скрипта на языке Python. Тем не менее, использование полного функционала графических ускорителей труднее осуществить указанным образом, что объясняется особенностями работы потоковых ядер. Использование алгоритмического языка Python в высокопроизводительных вычислениях во многих случаях значительно расширяет удобство создания, отладки и тестирования программного кода.
Ключевые слова: высокопроизводительные вычисления, суперкомпьютер, компьютерный кластер, язык программирования Python, CUDA

Primary author: ROMANOV, Alexander (Voronezh State University)

Co-authors: PETRISHCHEV, Konstantin (Voronezh State University); BORZUNOV, Sergei (Voronezh State University); KURGALIN, Sergei (Voronezh State University)

Presenter: ROMANOV, Alexander (Voronezh State University)

Session Classification: HPC

Track Classification: HPC