به نام خدا

دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)

درس پردازش دادگان حجیم

پیاده‌سازی و بهینه‌سازی ضرب ماتریس بر روی GPU با CUDA

استاد : دکتر پورفرد

سرپرست تیم استادیاری: دکتر طالبی

استادیار طراح تمرین: امیرمحمد تلیسچیان تبریزی

تمرین شماره دوم

پاییز ۱۴۰۴

هدف تمرین

هدف از این تمرین آشنایی دانشجویان با مبانی محاسبات موازی بر روی GPU و مدل برنامه‌نویسی CUDA است. در پایان این تمرین، انتظار می‌رود دانشجو بتواند:

۱. ساختار سلسله‌مراتبی اجرای برنامه در GPU را (Grid ، Block، Thread) درک کند.

۲. نحوه تخصیص و انتقال داده‌ها بین حافظه میزبان (Host) و GPU (Device) را یاد بگیرد.

۳. یک کرنل ساده برای محاسبه ضرب ماتریس بنویسد.

۴. از حافظه اشتراکی (Shared Memory) برای بهینه‌سازی عملکرد استفاده کند.

۵. سرعت اجرای GPU و CPU را مقایسه و تحلیل کند.

پیش‌نیازها

- آشنایی مقدماتی با زبان C یا++C
- آشنایی مقدماتی با مفاهیم موازی‌سازی
- نصب NVIDIA CUDA Toolkit و در اختیار داشتن GPU سازگار باCUDA

تمرین دوم

پردازنده‌های گرافیکی (GPU) برخلافCPU ، دارای تعداد زیادی هستهٔ محاسباتی هستند که برای اجرای هزاران نخ (thread)به‌صورت هم‌زمان طراحی شده‌اند. این ویژگی باعث می‌شود الگوریتم‌هایی که شامل عملیات تکراری و مستقل هستند، با سرعت بسیار بیشتری روی GPU اجرا شوند.

یکی از الگوریتم‌های پایه در علوم مهندسی و یادگیری ماشین، **ضرب ماتریس‌ها** است. این عملیات نمونه‌ای عالی برای درک نحوه موازی‌سازی داده‌ای است، زیرا هر عنصر از ماتریس خروجی را می‌توان مستقل از سایر عناصر محاسبه کرد.

اگر دو ماتریس A و B با اندازه N * N داده شده باشد , ضرب آنها به صورت زیر تعریف میشود :

$$\sum_{0}^{N-1} A_{ik} \,.\, B_{kj} = C_{ij}$$

بخش اول : پیاده‌سازی روی CPU

در این بخش ابتدا لازم است نسخه پایه ضرب ماتریس را روی CPU بنویسید تا بتوانید درستی نتایج GPU را با آن مقایسه کنید.

بخش دوم : پیاده‌سازی نسخه ساده GPU

در این بخش لازم است تا نسخه ای ساده از کرنل برای پیاده سازی این ضرب بنویسید که در آن هر thread یک درایه از ماتریس خروجی را محاسبه میکند.

در این نسخه هر thread چندین بار از حافظه global میخواند.

بخش سوم : بهینه‌سازی با Shared Memory (Tiled Multiplication)

در نسخه ساده، هر thread چندین بار از حافظه global می‌خواند.
در نسخهtiled ، داده‌ها در بلوک‌های کوچک‌تر در حافظه اشتراکی ذخیره می‌شوند تا دسترسی به حافظه سریع‌تر شود.

بخش چهارم : تحلیل و ارزیابی عملکرد

از توابع CUDA Event برای اندازه‌گیری زمان اجرای کرنل و کل برنامه استفاده کنید:

تحویل تمرین :

۱. کدهای کامل نسخهCPU ، GPUساده و GPU بهینه
۲. گزارش PDF شامل:
   - توضیح روش کار و مفاهیم استفاده‌شده
   - جداول زمان اجرا
   - تحلیل و نتیجه‌گیری نهایی
۳. فایل نتایج اجرا و خروجی‌ها

نکات :

- برای جلوگیری از خطای حافظه، همواره پس از اجرای کرنل از `cudaDeviceSynchronize()` استفاده کنید.
- حافظه‌های تخصیص‌یافته در GPU را با `cudaFree()` آزاد کنید.
- در گزارش، اختلاف زمانی بین نسخه‌ها را تحلیل کنید و علت فنی آن را توضیح دهید.

پیوست

برنامه‌های CUDA برخلاف برنامه‌های معمولی C/C++ توسط کامپایلر خاصی به نام nvcc کامپایل میشوند. کامپایلر nvcc بخشی از بسته‌ی NVIDIA CUDA Toolkit است و باید قبل از اجرای تمرین نصب شده باشد.

۱. بررسی نصب CUDA

برای اطمینان از نصب صحیح CUDA ، در خط فرمان (Terminal) دستور زیر را اجرا کنید:

```
~$ nvcc --version
```

اگر خروجی داشت به این معناست که به درستی نصب شده است.

۲. کامپایل برنامه با nvcc

فرض کنید شما فایل هایی با نام های زیر دارید :

- matrixmul.cu : این فایل شامل کرنل های شماست.
- main.c/cpp : این فایل شامل تابع main و فراخوانی کرنل هاست.

حال برای کامپایل این فایل ها باید دستور زیر در ترمینال وارد کنید :

```
~$ nvcc -arch=sm-80 main.cpp matrixmul.cu -o matrixmul
```

برای هر کارت گرافیک یک معماری داریم که باید با توجه به نوع کارت گرافیکی که از آن استفاده میکنید آنرا به درستی تنظیم کنید. در اینجا sm-80 مربوط به کارت گرافیک A100 میباشد.

۳. اجرای برنامه

با وارد کردن دستور زیر برنامه را میتوانید اجرا کنید :

```
~$ ./matrixmul
```

۴. ابزار بررسی GPU

با دستور زیر میتوانید نوع GPU و وضعیت آنرا مشاهده کنید :

```
$ nvidia-smi
```

۵. عیب یابی و خطاهای متداول

خطا	علت	راه حل
nvcc: command not found	CUDA Toolkit نصب نشده یا PATH تنظیم نیست	مسیر /usr/local/cuda/bin را به PATH اضافه کنید
undefined reference to cudaMalloc	با g++ کامپایل شده نه nvcc	همیشه با nvcc کامپایل کنید
cudaErrorInvalidDeviceFunction	معماری اشتباه GPU در زمان کامپایل	استفاده از arch=sm_ متناسب با کارت
cudaMemcpy failed	اندازه حافظه اشتباه یا ناهماهنگ	بررسی مقدار N و تخصیص حافظه
خروجی اشتباه	thread indexing یا sync اشتباه	بررسی ایندکس‌ها و استفاده از syncthreads

۶. منابع

کتاب ***Programming Massively Parallel Processors: A Hands-on Approach***

ویدیوهای برنامه نویسی موازی دانشگاه ایلینویز

https://youtube.com/playlist?list=PLRRuQYjFhpmvu5ODQoY2l7D0ADgWEcYAX&si=YWEh9LvMgsrReBcE