به نام خدا

برنامه‌نویسی چندهسته‌ای

دستور کار آزمایشگاه ۷

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

مقدمه

در آزمایش‌های قبلی با هیستوگرام و انواع پیاده‌سازی آن آشنا شدید. این پیاده‌سازی‌ها بر اساس داده‌های ورودی، سطل‌ها[1] خروجی و بر اساس هیستوگرام‌های محلی و روی پردازنده‌ی اصلی بوده است. همان‌طور که می‌دانید هیستوگرام یک برنامه‌ی حافظه[2] محور است و نیاز به دسترسی‌های زیادی به حافظه دارد. به همین جهت بهینه‌سازی و پوشش تأخیر دسترسی‌ها به حافظه یکی از مهم‌ترین رویکردها جهت بهبود سرعت انجام کار است. از جهتی دیگر، در آزمایش قبل با اثر اشغال[3] روی سرعت انجام کار آشنا شده‌اید. در این آزمایش علاوه بر آشنایی با جریان[4]ها، با بهینه‌سازی بیشتر دسترسی‌های حافظه و اثر منفی افزایش تصرف روی سرعت انجام کارهای حافظه محور آشنا خواهید شد.

اهداف آزمایش

- آشنایی با اثر منفی تصرف در کارهای حافظه محور
- آشنایی با حافظه‌ی پین‌شده[5]
- آشنایی با جریان
- اثر تصرف در اجرای هم‌زمان جریان‌ها

جهت انجام این آزمایش فایل histogram.cu در اختیار شما قرار گرفته‌است. در این فایل کد به دست آوردن هیستوگرام روی پردازنده‌ی گرافیکی و قسمت موردنیاز برای پر کردن آرایه‌ی موردنظر نوشته‌شده است و شامل متغیرهای زیر می‌شود و استفاده‌ی هر یک به شرح زیر است

```
#define MAX_HISTORGRAM_NUMBER 10000
#define ARRAY_SIZE 102400000
#define CHUNK_SIZE 100
#define SCALER 80
#define THREAD_COUNT 512
```

- MAX_HISTOGRAM_NUMBER: برای مشخص کردن محدوده‌ی اعداد موجود یا به‌عبارت‌دیگر، تعیین تعداد سطل‌های هیستوگرام استفاده می‌شود
- ARRAY_SIZE: تعداد کل عناصر مورد بررسی را مشخص می‌کند
- CHUNK_SIZE: تعداد اعضای موردبررسی توسط یک نخ را مشخص می‌کند
- SCALER: افزایش‌دهنده‌ی میزان کار هر نخ با انجام کارهای تکراری. از این متغیر برای طولانی کردن زمان اجرای تابع هسته به‌اندازه‌ای که مقادیر زمانی به‌دست‌آمده قابل استناد و برای نمونه‌گیری Visual profiler طولانی باشد. برای معماری پاسکال عدد مناسب ۸۰ است.
- THREAD_COUNT: تعداد نخ‌های هر بلوک را مشخص می‌کند

[1] bucket
[2] Memory intensive
[3] Occupancy
[4] Streams
[5] Pinned memory

قسمت اول: بررسی اثر تصرف بر میزان تسریع

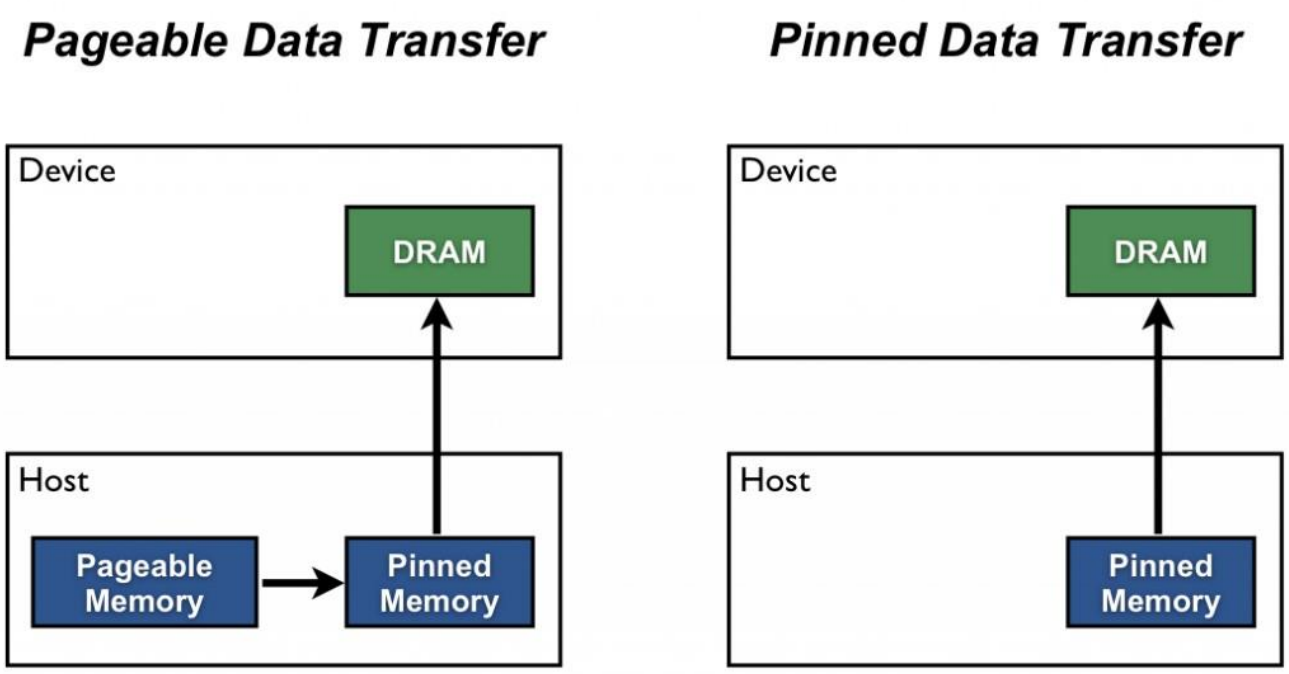
تابع هسته همراه کد سمت پردازنده‌ی اصلی در فایل histogram.cu موجود است. ابتدا کد را مطالعه کنید و پس از اطمینان از درک درست آزمایش با توجه به‌اندازه‌ی آرایه‌ی ورودی و تغییر THREAD_COUNT و CHUNK_SIZE تعداد بلوک‌های موردنیاز برای فراخوانی تابع را به دست آورید و جدول را کامل کنید و سؤالات زیر را پاسخ دهید. اندازه‌ی آرایه ورودی و بازه‌ی اعداد موردنظر در قسمت بالا معرفی‌شده است.

۱. دلیل اثر منفی افزایش اشغال در زمان اجرا چیست؟
۲. چگونه می‌توان با کم نکردن اشغال زمان اجرا را بهبود داد؟

تعداد نخ	۸	۱۶	۳۲	۲۵۶	۱۰۲۴
متغیر SCALER	80	80	80	80	80
تعداد بلوک					
اشغال نظری					
اشغال به‌دست‌آمده					
زمان اجرای تابع هسته					

قسمت دوم: استفاده از حافظه‌ی پین‌شده

در برنامه‌نویسی کودا به‌طور پیش‌فرض تخصیص حافظه برای پردازنده به‌صورت pageable است و پردازنده‌ی گرافیکی امکان دسترسی مستقیم به این نوع حافظه را ندارد. پس هنگام انتقال داده از حافظه‌ی pageable، ابتدا درایور CUDA حافظه‌ی page-locked یا پین‌شده را موقتاً تخصیص می‌دهد سپس آرایه‌ی موجود در حافظه‌ی pageable را به این حافظه منتقل می‌کند و سپس از این حافظه به حافظه‌ی پردازنده‌ی گرافیکی منتقل می‌کند. در شکل زیر این فرایند را می‌توانید ببینید.



شکل(۱): تفاوت حافظه پین‌شده و pageable

برای تخصیص حافظه به‌صورت پین‌شده، میانای برنامه‌ی کاربردی[6] کودا توابعی را در اختیار برنامه‌نویس قرار می‌دهد. یکی از این توابع، تابع cudaMallocHost است؛ که از نظر ساختاری بسیار شبیه cudaMalloc است اما حافظه تخصیص‌یافته روی حافظه‌ی پردازنده است.

در این قسمت آزمایش، با انجام تغییرات مناسب در کد حافظه را به‌صورت پین‌شده تخصیص دهید و زمان اجرای برنامه را در حالت زیر با حالت متناظر در قسمت قبل مقایسه کنید. سپس جواب سؤالات زیر را بدهید

۱. تفاوت حافظه Pageable با حافظه پین‌شده در چیست؟
۲. روش‌های دیگر تخصیص حافظه‌ی پین‌شده را معرفی کنید و تفاوت آن‌ها را با یکدیگر بیان کنید.
۳. دلیل دیگر بهبود زمان اجرا در استفاده از این روش چیست؟(به‌جز حذف حافظه‌ی پین‌شده موقت و زمان انتقال داده به آن)

قسمت سوم: استفاده از جریان‌ها و اثر اشغال در اجرای هم‌زمان آن‌ها

در این قسمت با توجه به آشنایی با محدودیت موجود در قسمت اول، به رویکرد جدیدی برای بهبود زمان اجرا نیاز داریم. یکی از این رویکردها، استفاده از جریان‌ها است. با انجام تغییرات مناسب حلقه‌ی خارجی که مربوط به SCALER است را بین جریان‌های مختلف تقسیم کنید و زمان اجرای تابع هسته را به دست آورید و جدول زیر را کامل کنید. سپس به سؤالات زیر پاسخ دهید. برای مشاهده‌ی اجرای هم‌زمان جریان‌ها از Visual Profiler استفاده کنید.

تعداد نخ	۸	۱۶	۳۲	۲۵۶	۱۰۲۴	۸	۱۶	۳۲	۲۵۶	۱۰۲۴
تعداد جریان	۲	۲	۲	۲	۲	۴	۴	۴	۴	۴
متغیر SCALER	۲۰	۲۰	۲۰	۲۰	۲۰	۲۰	۲۰	۲۰	۲۰	۲۰
تعداد بلوک										
اندازه‌ی تکه										
اشغال نظری										
جریان‌های هم‌زمان										
اشغال به‌دست‌آمده										
زمان اجرای تابع هسته										

۱. به نظر شما مسائل مناسب برای استفاده از جریان‌ها چه ساختاری دارند؟
۲. پردازنده‌ی گرافیکی اولویت را به حفظ حداکثر اشغال می‌دهد یا اجرای هم‌زمان حداکثری جریان‌ها؟
۳. آیا با تغییر کامل رویکرد و اختصاص هر جریان به شمردن یک یا چند نوع عضو از آرایه‌ی اصلی می‌توان زمان اجرا را بهبود داد؟ جواب خود را با دلیل توضیح دهید. (پیاده‌سازی موردنیاز برای این سؤال امتیازی است)

[6] Application Programming Interface (API)