فاز دوم پروژه برنامه نویسی چند هسته ای

ریحانه یگانه و زهرا حسینی

تصحیح کد سریال

از آنجا که در کد سریال به زبان c++ در کودا محدودیت هایی داریم و امکان پیاده سازی برخی موارد در این نوع زبان نیست (مانند کلاس و...) ابتدای کار کد را به زبان سی تبدیل کردیم که البته این تغییرات زمانبر و مشکل ساز بود.

موازی سازی در سطح GPU

مواردی که بابت موازی سازی با کودا به کد با توجه به شرایط ایجاد شده است شامل :

نسخه اولیه:

- استفاده از حافظه constant برای ذخیره کلید های رمزنگاری و s_box جهت دسترسی سریع تر به داده ها
- بازنویسی توابع برای اجرا روی GPU
- استفاده از کرنل برای رمزنگاری چندین بلاک (در نسخه ابتدایی)
  - در کد سریال بلاک های ۱۶ بایتی به ترتیب پردازش می شدند و برای تعداد بلاک های بالا، اجرای کد بسیار زمان بر میشد. در کد موازی با تعریف یک کرنل برای هر بلاک ۱۶ بایتی یک thread در GPU اختصاص دادیم که مسولیت رمزنگاری آن بلاک را دارد.
- استفاده از cudaMalloc و cudaMemcpy جهت مدیریت حافظه درGPU(درنسخه ابتدایی)
- استفاده از cudaEvent جهت اندازه گیری زمان اجرای کرنل و مقایسات بعدی

نسخه دوم:

- افزودن CUDA Streams

Stream ها به GPU این اجازه را می دهند که چندین عملیات را به صورت همزمان انجام دهند. در کد اولیه ما صرفا یک کرنل تغریف کرده بودیم که پرازش موازی رو انجام میداد اما در کد جدید با افزودن استریم ها شرایطی رو فراهم کردیم که چندین کرنل به صورت غیر همزمان اجرا شوند. بدین صورت که بلاک ها را بین جریان های تغریف شده (۴و۸و۱۶) تقسیم کردیم و سپس هر جریان برای بلاک های در اختیار قرار گرفته خودش یک کرنل اجرا می کند. انتقال داده ها فقط در جریان اول انچام شده است. با افزودن استریم ها زمان ما به طور قابل توجهی کاهش پیدا کرد.

- استفاده از cudaMemcpyAsync

ما از cudaMemcpyAsync استفاده کردیم تا انتقال داده به صورت غیر همزمان انجام شود و GPU بتواند همزمان با انتقال داده ها کرنل ها را نیز اجرا کند این کار باعث کاهش اجرای زمان کلی پردازش می شود.

!! در هنگام موازی سازی از طریق stream، مهم ترین مشکل سربار انجام این کار بود که نیاز به بالا رفتن throughput درآن اهمیت بالاتری پیدا می کرد.

نسخه سوم:

- پردازش چند بلاک توسط هر رشته

در نسخه پیشین هر رشته تنها یک بلاک را پردازش میکرد. اما در کد جدید تعیین کردیم که هر رشته بتواند چندین بلاک را پردازش کند. این موجب کاهش Grid size در کودا برای تعداد بلاک های بالا می شود و بهره وری را افزایش می دهد.

- استفاده از Pinned Memory
ما از cudaHostAlloc استفاده کردیم تا بتوانیم از حافظه پین شده استفاده کنیم. این حافظه به دلیل اینکه امکان انتقال مستقیم به GPU را فراهم می کند به اجرای ما سرعت می بخشد. همچنین برای آزادسازی این حافظه از cudaFreeHost استفاده کردیم.
- بهینه سازی توزیع بلاک ها
در کد جدید ابتدا یک تعداد پایه بلاک را بین استریم ها تقسیم کردیم و سپس بلاک های باقی مانده را بین استریم های اولیه تقسیم کردیم این توزیع نسبت به توزیع قبلی متعادل تر بوده.
- حذف انتقال داده ها صرفا توسط استریم ابتدایی
در کد جدید هر استریم داده های بلاک های خود را انتقال می دهد این کار موجب افزایش همزمانی می شود.

نسخه چهارم (حافظه مشترک):

برای بررسی تاثیر استفاده از حافظه مشترک بر روی زمان اجرا معماری را کمی تغییر دادیم . چالشی که وجود داشت این بود که GPUها معمولا 48KB حافظه مشترک در هر SM دارند و برای این که به محدودیت های حافظه نخوریم باید به حجم داده مشترک توجه میکردیم. 176 round_keys , 256 S_box بایت بودند و shared_state هم 2048 بایت بود تعداد ترد ها را به ۱۲۸ کاهش دادیم.

مقایسه زمان های اجرا و انتخاب بهترین نسخه:

زمان اجرای کد سریال:

100 input: Average: 0.973 ms | Average: 1.144 ms | Average: 1.238 ms

1000 input: Average: 14.896 ms | Average: 12.121 ms | Average: 14.711 ms

10000 input: Average: 212.603 ms | Average: 186.329 ms | Average: 172.137 ms

همانطور که مشاهده می شود حدودا از ۱۰۰۰۰ ورودی به بعد زمان ما به صورت قابل توجهی افزایش می یابد به همین دلیل نقطه شروع را این مورد در نظر گرفتیم.

زمان اجرای کد قبل از بهبود استریمینگ:

10000 input:

4 streams: Average: 2.481 ms | Average: 2.522 ms | Average: 2.497 ms |

8 streams: Average: 4.938 ms | Average: 4.864 ms | Average: 4.856 ms |

16 streams: Average: 6.547 ms |

``در این تعداد ورودی مشاهده می شود با افزایش تعداد استریم از ۸ به بعد ما سربار داریم و زمان اجرای ما افزایش می یابد.

زمان اجرای کد پس از بهبود استریمینگ:

10000 input:

4 streams: Average: 0.653 ms | Average: 0.685 ms |

8 streams: Average: 0.723 ms | Average: 0.717 ms |

16 streams: Average: 0.687 ms | Average: 0.597 ms |

همانطور که مشاهده میشود عملکرد کد به شدت بهبود یافته و زمان های اجرا به طرز قابل توجهی کاهش یافته اند پس از این مرحله به سراغ بررسی تاثیر حافظه ها بر روی زمان اجرا رفتیم. کدی که تا گذشته ران می شد برای حافظه های non_unified طراحی شده بود . کد مجدد مورد بررسی و تصحیح قرار گرفت تا ببینیم در صورت وجود معماری حافظه مشترک چه تاثیری بر زمان اجرا می گذارد.

زمان اجرای کد برای حافظه های مشترک:

10000 input:

4 stream: Average: 0.157 ms | Average: 0.162 ms |

8 stream: Average: 0.212 ms | Average: 0.209 ms |

16 stream: Average: 0.289 ms | Average: 0.273 ms |

همانطور که مشاهده می شود زمان اجرا به طور قابل توجهی کاهش یافته است تقریبا زمان اجرا ۱/۴ شده ایت که نشان می دهد در معماری حافظه مشترک به این دلیل که داده ها یک بار به حافظه مشترک کپی می شوند و دسترسی به آن ها سریع تر است زمان اجرا به طور خاصی کاهش می یابد که البته این مورد برای الگوریتم رمز نگاری ما که ترد ها مکررا از s_box استفاده می کنند تاثیر قابل توجهی گذاشته است.

بررسی تاثیر پارامتر های مختلف در زمان پردازش

Thread Block Size:

این مورد به تعداد ترد ها در هر بلاک کودا مربوط است به معنای دیگر با تنظیم این مورد تعیین میکنیم چند ترد به صورت همزمان در یک بلاک اجرا شوند. ما اعداد ۶۴، ۱۲۸، ۱۹۲ و ۲۵۶ را برای آزمایش انتخاب کردیم.

Kernel Granularity:

blocks_per_thread این پارامتر تعیین می کند هر ترد چند بلاک AES را پردازش می کند اگر این مورد و افزایش بدیم تعداد ترد های کمتری نیاز خواهیم داشت اما ممکنه SM ها بیکار بمونن و اگر مقدارش رو خیلی کم انتخاب کنیم باعث سربار میشه چون ترد ها و مدیریتشون زمان بر میشه. ما اعداد ۱، ۲، ۴ و ۸ رو برای آزمایش انتخاب کردیم.

```
Reading plaintexts from plaintext_10000_blocks.txt
Successfully read 10000 keys from plaintext_10000_blocks.txt
Reading ciphertexts from ciphertexts_10000_blocks.txt
Successfully read 10000 keys from ciphertexts_10000_blocks.txt

Testing different THREADS_PER_BLOCK and blocks_per_thread combinations:
------------------------------------------------------------
| THREADS_PER_BLOCK | blocks_per_thread | Avg Time (ms) | Valid |
------------------------------------------------------------
|                64 |                 1 |         0.245 |   Yes |
|                64 |                 2 |         0.249 |   Yes |
|                64 |                 4 |         0.285 |   Yes |
|                64 |                 8 |         0.409 |   Yes |
|               128 |                 1 |         0.226 |   Yes |
|               128 |                 2 |         0.249 |   Yes |
|               128 |                 4 |         0.301 |   Yes |
|               128 |                 8 |         0.397 |   Yes |
|               192 |                 1 |         0.245 |   Yes |
|               192 |                 2 |         0.258 |   Yes |
|               192 |                 4 |         0.315 |   Yes |
|               192 |                 8 |         0.397 |   Yes |
|               256 |                 1 |         0.234 |   Yes |
|               256 |                 2 |         0.271 |   Yes |
|               256 |                 4 |         0.313 |   Yes |
|               256 |                 8 |         0.398 |   Yes |
------------------------------------------------------------
Best configuration:
THREADS_PER_BLOCK = 128, blocks_per_thread = 1, Avg Time = 0.226 ms
```

```
Reading plaintexts from plaintext_100000_blocks.txt
Successfully read 100000 keys from plaintext_100000_blocks.txt
Reading ciphertexts from ciphertexts_100000_blocks.txt
Successfully read 100000 keys from ciphertexts_100000_blocks.txt

Testing different THREADS_PER_BLOCK and blocks_per_thread combinations:
------------------------------------------------------------
| THREADS_PER_BLOCK | blocks_per_thread | Avg Time (ms) | Valid |
------------------------------------------------------------
|                64 |                 1 |         0.607 |   Yes |
|                64 |                 2 |         0.653 |   Yes |
|                64 |                 4 |         0.777 |   Yes |
|                64 |                 8 |         1.013 |   Yes |
|               128 |                 1 |         0.596 |   Yes |
|               128 |                 2 |         0.659 |   Yes |
|               128 |                 4 |         0.825 |   Yes |
|               128 |                 8 |         1.098 |   Yes |
|               192 |                 1 |         0.608 |   Yes |
|               192 |                 2 |         0.683 |   Yes |
|               192 |                 4 |         0.823 |   Yes |
|               192 |                 8 |         1.139 |   Yes |
|               256 |                 1 |         0.599 |   Yes |
|               256 |                 2 |         0.707 |   Yes |
|               256 |                 4 |         0.890 |   Yes |
|               256 |                 8 |         1.258 |   Yes |
------------------------------------------------------------
Best configuration:
THREADS_PER_BLOCK = 128, blocks_per_thread = 1, Avg Time = 0.596 ms
```

اگر تعداد بلاک ها برای هر ترد را افزایش دهیم در نتیجه هر ترد باید به صورت سریال این بلاک ها را پیمایش کند که در نتیجه با افزایش تعداد بلاک هایی که به هر ترد اختصاص می دهیم چون به صورت سریال این مورد د رحال انجام است، زمان ما افزایش می یابد.

در عین حال اگر تعداد ترد ها در هر بلاک کودا را افزایش دهیم عملکرد بهتر می شود اما این مورد فقط تا عدد ۱۲۸ به خوبی عمل می کند و بیشتر از آن سربار دارد.

همچنین به دلیل وجود استریم ها، استریم ها را نیز افزایش دادیم و زمان های اجرا را مقایسه کردیم. استریم ۱۶ و ۳۲ زمان های به نسبت بهتری دادند و از استریم ۶۴ ما شاهد افزایش زمان اجرا بودیم که نشان دهنده افزایش سربار بود چون تعداد کمی بلاک به هر استریم می رسید.

```
Reading plaintexts from plaintext_100000_blocks.txt
Successfully read 100000 keys from plaintext_100000_blocks.txt
Reading ciphertexts from ciphertexts_100000_blocks.txt
Successfully read 100000 keys from ciphertexts_100000_blocks.txt

Testing different THREADS_PER_BLOCK and blocks_per_thread combinations:
------------------------------------------------------------
| THREADS_PER_BLOCK | blocks_per_thread | Avg Time (ms) | Valid |
------------------------------------------------------------
|                64 |                 1 |         0.605 |   Yes |
|                64 |                 2 |         0.677 |   Yes |
|                64 |                 4 |         0.817 |   Yes |
|                64 |                 8 |         1.025 |   Yes |
|               128 |                 1 |         0.597 |   Yes |
|               128 |                 2 |         0.688 |   Yes |
|               128 |                 4 |         0.853 |   Yes |
|               128 |                 8 |         1.160 |   Yes |
|               192 |                 1 |         0.605 |   Yes |
|               192 |                 2 |         0.691 |   Yes |
|               192 |                 4 |         0.853 |   Yes |
|               192 |                 8 |         1.143 |   Yes |
|               256 |                 1 |         0.609 |   Yes |
|               256 |                 2 |         0.712 |   Yes |
|               256 |                 4 |         0.937 |   Yes |
|               256 |                 8 |         1.403 |   Yes |
------------------------------------------------------------
Best configuration:
THREADS_PER_BLOCK = 128, blocks_per_thread = 1, Avg Time = 0.597 ms
```

```
Reading plaintexts from plaintext_100000_blocks.txt
Successfully read 100000 keys from plaintext_100000_blocks.txt
Reading ciphertexts from ciphertexts_100000_blocks.txt
Successfully read 100000 keys from ciphertexts_100000_blocks.txt

Testing different THREADS_PER_BLOCK and blocks_per_thread combinations:
------------------------------------------------------------
| THREADS_PER_BLOCK | blocks_per_thread | Avg Time (ms) | Valid |
------------------------------------------------------------
|                64 |                 1 |         0.616 |   Yes |
|                64 |                 2 |         0.664 |   Yes |
|                64 |                 4 |         0.770 |   Yes |
|                64 |                 8 |         0.952 |   Yes |
|               128 |                 1 |         0.607 |   Yes |
|               128 |                 2 |         0.657 |   Yes |
|               128 |                 4 |         0.777 |   Yes |
|               128 |                 8 |         1.088 |   Yes |
|               192 |                 1 |         0.616 |   Yes |
|               192 |                 2 |         0.679 |   Yes |
|               192 |                 4 |         0.832 |   Yes |
|               192 |                 8 |         1.148 |   Yes |
|               256 |                 1 |         0.606 |   Yes |
|               256 |                 2 |         0.658 |   Yes |
|               256 |                 4 |         0.892 |   Yes |
|               256 |                 8 |         1.124 |   Yes |
------------------------------------------------------------
Best configuration:
THREADS_PER_BLOCK = 256, blocks_per_thread = 1, Avg Time = 0.606 ms
```