به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

High Performance Regex Matching

تمرین شماره ۳ درس «پردازش چندهسته‌ای»

استاد: سید مهدی ابراهیمی

نیمسال دوم ۱۴۰۴-۱۴۰۳

مقدمه

در دنیای امروز که حجم اطلاعات به‌طرز چشمگیری افزایش یافته و هر فرد روزانه مقادیر زیادی داده تولید می‌کند، لازم است این حجم عظیم داده به‌سرعت پردازش شوند. برای مثال، پیش از ارسال کامل بسته‌های شبکه (packet) یا تأیید تراکنش‌های بانکی، باید محتوای آن‌ها با سرعت بالا بررسی شود. در این شرایط، ما به راهکارهایی نیاز داریم که بتوانند روی داده‌های بسیار حجیم، پردازش‌های پیچیده را در زمان کوتاه اجرا کنند. در این تمرین، قصد داریم به یکی از این چالش‌ها و ابزارهای مرتبط با آن بپردازیم.

Regex چیست؟ قدرتی برای یافتن الگوها در متن

عبارات با قاعده، یا به اختصار Regex (مخفف Regular Expressions)، ابزاری قدرتمند و انعطاف‌پذیر برای جستجو، یافتن و دستکاری الگوهای متنی هستند. این "الگوها" مجموعه‌ای از کاراکترها یا قواعدی خاص‌اند که شما تعریف می‌کنید تا دقیقاً آنچه را که در یک رشته متنی به دنبالش هستید، پیدا کنید.

فرض کنید به دنبال تمام شماره تلفن‌ها یا آدرس‌های ایمیل در یک فایل بسیار بزرگ هستید؛ Regex به شما این امکان را می‌دهد که این الگوها را با زبانی فشرده و کارآمد توصیف کنید. موتور Regex سپس متن را اسکن کرده و تمام موارد منطبق با الگوی شما را شناسایی می‌کند. اساس کار Regex بر پایه نظریه اتوماتای متناهی (Finite Automata) در علوم کامپیوتر است. این به معنای آن است که هر الگوی Regex می‌تواند به یک ماشین حالت (State Machine) تبدیل شود که متن را قدم به قدم پردازش می‌کند.

به دلیل توانایی Regex در توصیف الگوهای پیچیده و کاربرد گسترده آن در پردازش متن، اعتبارسنجی ورودی‌ها، و استخراج اطلاعات در بسیاری از زبان‌های برنامه‌نویسی، درک آن کلیدی برای پردازش داده‌های متنی است. در این تمرین به بررسی عملکرد Regex Matcher های موجود روی دو بستر CPU و GPU میپردازیم.

CPU-Based Regex Matching (Hyperscan)

Hyperscan یک کتابخانه بسیار پرسرعت برای پیدا کردن الگوهای متنی (مثل الگوهای تعریف شده با Regex) در حجم زیادی از اطلاعات است. این ابزار که توسط اینتل ساخته شده، به جای اینکه الگوها را تک‌تک و پشت سر هم جستجو کند، می‌تواند همزمان و با سرعت بی‌نظیری، هزاران الگو را در حجم عظیمی از متن پیدا کند. می‌توانید از طریق این فایل و ویدیو بیشتر درمورد این کتابخانه مطالعه کنید.

شرح مسئله

هدف این تمرین، پیاده‌سازی یک سیستم کارآمد برای تطبیق پرسرعت حجم عظیمی از الگوهای Regex بر روی داده‌های متنی حجیم است. ما با استفاده از کتابخانه Hyperscan، که بهینه‌سازی ویژه‌ای برای این نوع پردازش‌ها دارد، به دنبال دستیابی به عملکرد بالا در سناریوهای واقعی خواهیم بود.
در این تمرین، می‌خواهیم سیستمی را طراحی و پیاده‌سازی کنیم که قادر باشد ۲۰۰,۰۰۰ قانون Regex را به صورت همزمان بر روی یک فایل متنی حجیم اعمال کند. این فایل متنی شامل خطوط متعددی است که هر یک تقریباً ۲ کیلوبایت حجم دارند و رشته‌هایی از حروف را شامل می‌شوندکه باید قوانین نام برده شده بر روی این ورودی‌ها چک شوند. هدف اصلی ما شمارش و گزارش‌دهی تعداد کل تطابق‌ها (Matches) با حداکثر سرعت ممکن است.
یکی از اهداف این تمرین ، تحلیل تاثیر استفاده از multi-threading در عملکرد کلی سیستم است. برای این منظور، برنامه با تعداد threadهای مختلف اجرا خواهد شد و عملکرد آن در سناریوهای مختلف اندازه‌گیری و ثبت می‌شود. سناریو مورد بررسی به صورت single-thread و multi-thread با تعداد thread های 1, 2, 4, 8, 16خواهد بود. توجه شود خروجی تطابق قوانین برای تعداد thread های مختلف باید یکسان باشد و تغییر تعداد thread تاثیری رو این خروجی نخواهد داشت.

معیارهای عملکردی که اندازه‌گیری خواهند شد:

۱. **Throughput (Data Input/sec):**
سرعت پردازش دیتای ورودی، بر حسب تعداد ورودی در ثانیه.

۲. **Throughput (MBytes/sec):**
سرعت پردازش دیتای ورودی, بر حسب حجم (MBytes) دیتا در ثانیه

۳. **Throughput (Match/sec):**
تعداد تطابق‌های (matches) پردازش‌شده در ثانیه.

٤. **Latency:**
میانگین مدت‌زمان لازم برای پردازش یک خط ورودی، بر حسب میلی‌ثانیه.

نوع فایل‌های ورودی

برای این تمرین، با دو نوع فایل ورودی سروکار داریم:

۱- فایل قوانین Regex (فرمت TXT): قوانین Regex از طریق یک فایل متنی ساده (TXT) فراهم می‌شود که هر خط آن شامل یک الگوی Regex است. شناسه (ID) هر قانون برابر با شماره خط آن (از 0 شروع) در فایل است.

شکل۱: چند مثال از قوانین Regex

```
/desulfurs.*selfishnesses/is

/error.*(fatal|critical)/

/\buser_\d+\b/i
```

فایل داده‌های متنی (فرمت TXT): این فایل شامل داده‌های اصلی است که قوانین Regex روی آن‌ها اعمال می‌شوند. این فایل از خطوط متعدد تشکیل شده است؛ هر خط یک ورودی مستقل برای بررسی الگوهای متنی می باشد ، که شامل رشته‌ای بلند از حروف و کاراکترها است. توجه شود که چندین فایل داده متنی به شما داده خواهد شد و نیاز است تمام آنها بررسی شود.

شکل۲: مثال از ورودی‌های داده شده

```
the user_123 triggered a fatal error in the system

desulfurs and selfishnesses are both uncommon terms

this line does not match any specific rule
```

خروجی مورد نظر

خروجی تطابق قوانین: پس از اجرای پردازش، خروجی نهایی به صورت یک فایل متنی (TXT) خواهد بود. هر خط از این فایل خروجی مربوط به یک ورودی در فایل داده‌ها است (به همان ترتیب و شماره خط). این خط‌ها شامل لیستی از شناسه‌های (ID) قوانین Regex هستند که با آن ورودی مچ شده‌اند. اگر هیچ قانونی با آن خط مچ نشود، خط مربوطه خالی خواهد بود. به ازای هر فایل داده ورودی باید یک فایل خروجی متناظر تولید شود.

شکل۳: مثال از خروجی مورد انتظار برنامه

```
0,1,4



3
```

در این مثال:

- ورودی دیتای اول با pattern های [0,1,4] مچ شده است.
- ورودی دیتای دوم با هیچ pattern ای مچ نشده است.
- ورودی دیتای سوم تنها با 3 مچ شده است.

گزارش تحلیل عملکرد (فرمت CSV): در این فایل csv, خروجی عملکردی برنامه بر اساس معیار های مطرح شده, به ازای تعداد thread های مختلف گزارش می شود. این فایل csv شامل ستون های زیر باید باشد:

- threads
- throughput_input_per_sec
- throughput_mbytes_per_sec
- throughput_match_per_sec
- latency

شکل۴: مثال از خروجی مورد انتظار در فایل csv

```
threads,throughput_input_per_sec,throughput_mbytes_per_sec,throughput_match_per_sec,latency
1,1200,2.34,5300,5.3
2,2400,4.58,10200,3.1
4,4600,8.76,18900,2.0
8,7800,14.9,31800,1.5
16,9100,17.3,37000,1.3
32,9200,17.5,37200,1.4
```

به ازای هر فایل داده ورودی باید یک فایل خروجی متناظر تولید شود.

GPU-Based Regex Matching

شرح مسئله

برای تکمیل تحلیل عملکرد تطابق الگوها (Regex Matching)، در این تمرین بخشی را به بررسی و مقایسه اجرای library های مربوطه بر پایه GPU اختصاص داده‌ایم. انتظار می رود استفاده از GPU، به‌ویژه در حجم بالای داده‌های متنی و تعداد زیاد قوانین، می‌تواند بهبود چشم‌گیری در سرعت پردازش به همراه داشته باشد. Regex matcherهای مختلفی بر بستر GPU توسعه داده شده است که هر یک مزیت ها و ایراداتی دارد. شما باید این library های مختلف را پیدا کرده و با مقایسه خروجی عملکردی آنها به این مزیت ها و ایرادات برسید. در صورت تمایل میتوانید به جای استفاده از library های توسعه داده شده, regex matcher خود را توسعه دهید و به بهبود عملکردی آن بپردازید. نکته مهم این در این بخش بررسی عملکرد GPU در اجرای این گونه مسایل می باشد.

فایل های ورودی / خروجی

فایل های ورودی این بخش دقیقا همان فایل های ورودی بخش CPU-Based می باشد. خروجی این بخش شامل یک فایل خروجی تطابق قوانین به ازای هر فایل داده متنی ورودی, دقیقا مشابه بخش قبلی و یک فایل گزارش تحلیل عملکرد به شرح زیر است:

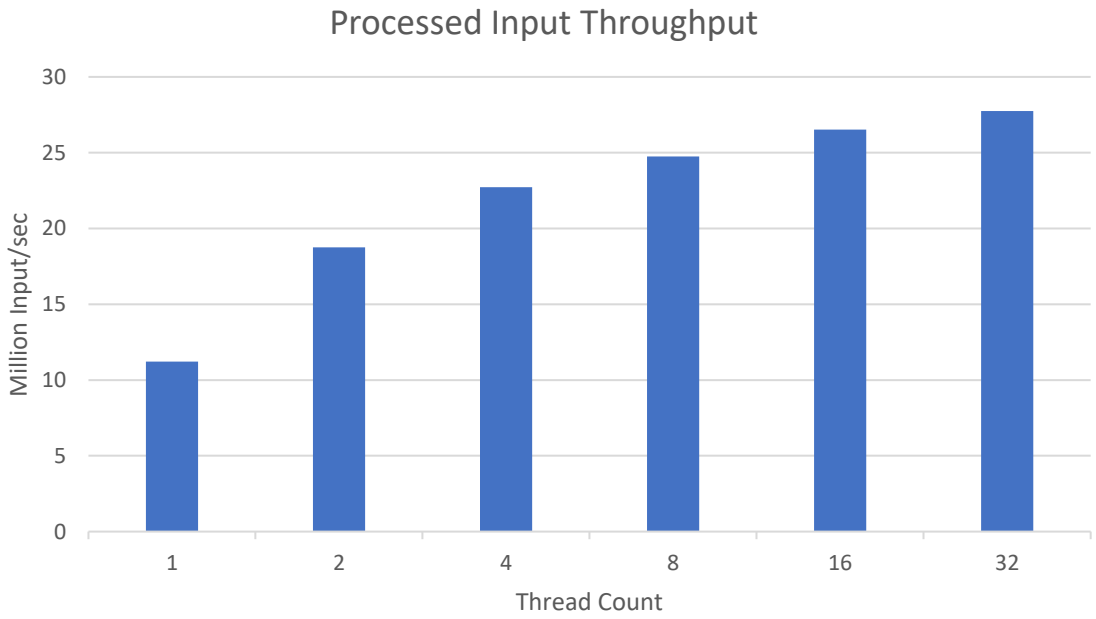
- معیار های عملکردی که باید گزارش شود دقیقا مشابه بخش قبلی است
- زمان اجرای اندازه گیری شده باید شامل هم بخش انتقال (HD-DH) و هم بخش پردازشی (Kernel) باشد.
- فایل خروجی به فرمت csv مشابه بخش قبلی بدون ستون threads است و به جای آن ستونی تحت عنوان matcher_name داریم که نام library استفاده شده می باشد. در صورت توسعه library توسط شما, این فایل صرفا شامل یک ردیف با نام انتخابی شما خواهد بود.

Final Performance Analysis

شرح بخش

در نهایت شما باید با نگاهی تحلیلی به بررسی عملکرد regex matcher های بررسی شده, مقایسه عملکرد دو بستر CPU/GPU در این گونه مسایل و بررسی بهبود های احتمالی بپردازید. نوع نگاه و استدلال شما در این تحلیل مورد ارزیابی قرار خواهد گرفت. چند نمونه بررسی پیشنهادی:

- در بخش CPU با درج کردن نمودار perf/threads بررسی کنید multi-thread کردن برنامه تا چه حدی میتولند به بهبود بی انجامد.
- پارامتر های تاثیر گذار بر روی عملکرد Hyperscan را بیابید.
- نتایج اجرا بر روی فایل های داده متنی ورودی متفاوت را مقایسه و تحلیل کنید. تحلیل‌ها باید شامل نمودار نتایج بدست آمده باشد.
- نمودارها باید به‌صورت زیر باشد.



نمودار شماره۱: نمودار تعداد داده‌های ورودی پردازش شده در واحد زمان برای تعداد نخ‌های متفاوت

- با درج کردن نمودار perf/library به مقایسه library های بررسی شده روی GPU بپردازید و با توضیح مختصر راجب نحوه پیاده سازی هر کدام, مزیت ها و ایرادات هر کدام را ذکر کنید.
- در صورت پیاده سازی روی GPU توسط شما, شرح مختصری از پیاده سازی و بهبود های انجام شده بیاورید.

- در نهایت با درنظر گرفتن بهترین عملکردی که از GPU مشاهده کرده اید, به مقایسه دو بستر CPU/GPU بپردازید و دلیل برتری هر کدام که بهتر بود (یا دلیل ضعف هر کدام که ضعیف تر بود) را ذکر کنید. در نهایت این تحلیل کمک می‌کند تا مشخص شود که برای بارهای کاری مختلف (حجم داده بالا، حساسیت به تأخیر، تعداد قوانین زیاد) کدام بستر مناسب‌تر است: Hyperscan بر بستر CPU یا اجرا بر بستر GPU.

قوانین:

- کد توسعه داده‌شده توسط شما می‌بایست توسط makefile قابل build باشد.
- کد شما صرفا توسط ماشین build می‌شود و می‌بایست توسط ماشین قابل build باشد.
- برنامه‌ی شما صرفا توسط ماشین اجرا می‌شود.
- نتایج برنامه‌ی شما می‌بایست الزاما در ساختار مشخص شده خروجی تولید کند. نتایج صرفا با ماشین بررسی می‌شوند.

ساختار فایل تحویلی

- برنامه‌ی شما توسط سامانه به‌صورت خودکار از حالت فشرده خارج خواهد شد. لطفاً فایل نهایی را با فرمت .tar.gz و با استفاده از دستور زیر ایجاد کنید:

```
tar -czf  HW3_MCC_030402_StudentID.tar.gz HW3_MCC_030402_StudentID
```

- ساختار پوشه‌ی داخلی باید دقیقاً به صورت زیر باشد:
  - پوشه‌ی HW3_MCC_030402_StudentID
    - پوشه‌ی bin شامل باینری شما
    - پوشه‌ی src شامل کدهای شما
    - پوشه‌ی results، شامل نتایج شما
  - برنامه‌ی باینری، می‌بایست با فرمت زیر تولید شود. صرفا توسط ماشین اجرا می‌شود.
    - HW3_MCC_030402_StudentID

ساختار تولید نتایج و فایل‌های خروجی

- نام فایل‌های نتایج باید مطابق الگوی زیر باشد:

```
Results_HW3_MCC_030402_StudentID_{CPU/GPU}_{DataSet}_{NumOfCPUThreads/GPULibrary}.csv
```

  - به‌عنوان مثال:

```
Results_ MCC_030402_StudentID_CPU_set1_1.csv
```

- تولید سایر فایل‌های لاگ (به جز فایل‌های نتایج) آزاد است و به دلخواه شما انجام می‌شود؛ این فایل‌ها در فرآیند تصحیح ملاک ارزیابی قرار نخواهند گرفت.

۱. مراحل انجام تمرین باید به‌صورت گزارش ارائه شود. گزارش باید شامل نتایج به‌دست آمده و سایر موارد خواسته شده به‌صورت ذکر شده درصورت تمرین باشد.
۲. الزامات فنی: حتما از زبان برنامه‌نویسی C استفاده کنید. همچنین کد شما باید روی سیستم عامل Ubuntu Linux (نسخه 22.04) کامپایل و اجرا شود.
۳. فایل‌ها، خروجی‌های به دست آمده (کد برنامه، library، makefile و نسخه باینری اپلیکیشن تست و ...) و فایل گزارش را به‌صورت فشرده با فرمت زیر در سامانه درس‌افزار (CW) بارگزاری نمایید.

HW3_MCC_030402_StudentID.tar.gz

۴. تاریخ تحویل تمرین ۳ شهریور است و این تاریخ به هیچ‌وجه تغییر نمی‌کند و به ازای هر روز تاخیر ٪۱۵ نمره را از دست خواهید داد و بعد از ۳ روز نمره این تمرین ۰ خواهد شد.

۵. می‌توانید سوالات یا ابهامات خود را به ایمیل s.yazdan566@gmail.com ارسال نمایید.

۶. رعایت آداب آموزشی در انجام پروژه و تمرین‌های درسی الزامی است. لطفا آیین‌نامه مصوب دانشکده (آداب‌نامه‌ی انجام تمرین‌های درسی) را دقیقا مطالعه فرمایید. در صورت مشاهده هرگونه تقلب علمی، نمره تمرین برای هر دو طرف ۱۰۰- منظور خواهد شد.

با آرزوی موفقیت