تمرین سوم: برنامه نویسی موازی با کمک POSIX

محمد هاشمی ۸۱۰۱۹۷۴۲۳ md.hashemi@ut.ac.ir
شاهین منتظری ۸۱۰۱۹۷۳۹۰ shahin.montazeri@ut.ac.ir

چکیده

کتابخانه پازیکس زبان سی به انگلیسی C POSIX library یک کتابخانه برنامه‌نویسی است که توسط استاندارد پازیکس برای زبان برنامه‌نویسی سی تعریف شده و از سیستم‌عامل‌های سازگار با این استاندارد انتظار می‌رود که این کتابخانه را فراهم کنند. این کتابخانه هم‌زمان با کتابخانه استاندارد C توسعه یافت.

کلمات کلیدی

C++، POSIX، mutex.

۱- مقدمه

تلاش‌هایی برای هماهنگ کردن کتابخانه پازیکس با کتابخانه سی انجام شده‌است. پازیکس توابعی اضافه بر سازمان از توابعی که در کتابخانه استاندارد سی وجود دارند، تعریف و معرفی کرده‌است. کتابخانه استاندارد سی نیز خود بخشی از کتابخانه پازیکس محسوب می‌شود. ریسه‌های پازیکس به انگلیسی: POSIX threads یا به اختصار pthreads اشاره به تعدادی رابط برنامه‌نویسی نرم‌افزار در استاندارد پازیکس دارد که برای کار با ریسه‌ها در نظر گرفته شده‌اند.در سیستم‌عامل‌های سازگار با این استاندارد مانندفری‌بی‌اس‌دی، اپن‌بی‌اس‌دی، نت‌بی‌اس‌دی، لینوکس، سولاریس و ...، پیاده‌سازی‌های مختلفی از این رابط‌های برنامه‌نویسی وجود دارد. این رابط‌های برنامه‌نویسی در یک فایل سرآیند به نام pthreads.h تعریف شده‌اند. این فایل حاوی تعداد تابع، نوع دادهو تعدادی ثوابت است. حدوداً صد عدد روال مختلف برای کار با ریسه‌ها فراهم شده که نام تمام آنها با pthread_ شروع می‌شود. این روال‌ها را می‌توان به چهار دسته تقسیم کرد : مدیریت ریسه‌ها - ایجاد، اتصال و ...mutex ها، متغیرهای شرطی و همگام‌سازی.

۲- تمرین اول: پیدا کردن بیشترین مقدار

در این تمرین هدف اصلی بدست آوردن تسریع محاسبات توسط یک برنامه موازی شده به وسیله POSIX و با کتابخانه Ptread می‌باشد. ساختار کلی این برنامه به این صورت است که یک آرایه با سایز بزرگ (در این مثال مقدار ۱۰۰۰۰۰) در اختیار ما گذاشته شده است که با مقادیر رندوم بین ۰ تا ۱۰۰ مقداردهی شده‌اند. هدف از این تمرین بدست آوردن بزرگترین عدد در میان تمامی خانه‌های این آرایه نسبتا طویل است. در ابتدا باید این برنامه را به صورت سریال و سپس با استفاده از دستورات POSIX به حالت موازی پیاده سازی کنیم. باید توجه داشت که این موازی سازی سرباری را به مدار ما اضافه می‌کند. ازین رو با توجه به تعداد نسبتا کم ۱۰۰۰۰۰۰، تسریع کمی را نسبت به حالت اجرای سریال آن دریافت می‌کنیم زیرا سربار سیستم برای ایجاد threadها و مدیریت آن‌ها بسیار بیشتر از تسریع بدست آمده توسط این سیستم است این در حالی است که با افزایش تعداد خانه‌های آرایه به اعدادی چون یک میلیون میتوانیم تسریع بهتری بدست آوریم.

این برنامه را پنج بار اجرا می‌کنیم. در شکل ۱ نتایج برنامه برای یکی از اجراهای آرایه ۱۰۰۰۰۰ خانه‌ای و در شکل ۲ نتایج برنامه برای یکی از اجراهای آرایه ۱۰۰۰۰۰۰ خانه‌ای را مشاهده می‌کنید. در جدول ۱ و ۲ نیز زمان پنج بار اجرای دو برنامه فوق و تسریع اجرای آن‌ها را مشاهده می‌کنید.

```
Execution time of Serial code: 0.000104603 seconds
99.999992,
Execution time of POSIX code: 0.000313416 seconds
99.999992 ,
Speedup from Serial to POSIX = 0.33
```

شکل (۱): نتایج برنامه پیدا کردن بیشترین مقدار آرایه‌ای با ۱۰۰۰۰۰ خانه

```
Execution time of Serial code: 0.00103222 seconds
99.999992,
Execution time of POSIX code: 0.000608674 seconds
99.999992 ,
Speedup from Serial to POSIX = 1.70
```

شکل (۲): نتایج برنامه پیدا کردن بیشترین مقدار آرایه‌ای با ۱۰۰۰۰۰ خانه

جدول (۱): زمان اجرای بیشترین مقدار ۱۰۰۰۰۰ ورودی

زمان اجرای سریال	زمان اجرای موازی	میزان بهبود یافته (n برابر)	میانگین بهبود یافته
۰,۱۰۴	۰,۳۱۳	۰,۳۳	۰,۳۷
۰,۱۳۴	۰,۴۳۸	۰,۳۱	
۰,۱۲۸	۰,۴۶۲	۰,۲۸	
۰,۳۰۲	۰,۴۷۰	۰,۶۴	
۰,۱۱۸	۰,۴۲۸	۰,۲۸	

جدول (۲): زمان اجرای بیشترین مقدار ۱۰۰۰۰۰۰ ورودی

زمان اجرای سریال	زمان اجرای موازی	میزان بهبود یافته (n برابر)	میانگین بهبود یافته
۱,۰۳۲	۰,۶۰۸	۱,۷۰	۱,۶۴
۱,۳۹۹	۰,۷۴۳	۱,۸۸	
۲,۵۷۴	۰,۱۰۶	۲,۴۱	
۱,۰۴۸	۱,۰۳۰	۱,۰۲	
۱,۰۳۵	۰,۸۶۲	۱,۲۰	

در نمونه کد ۱ میتوانید بخش مربوط به موازی‌سازی این برنامه را مشاهده کنید.

```
//Parallel
void *task(void *arg)
{
    int *incoming_param = (int *)arg;
    float *result = (float *)malloc(sizeof(float));
    int i;
    for (i = *incoming_param * (Array_Size / NUM_THREADS); i<(*incoming
        if (*(a + i ) > *result)
            *result = *(a + i );

    return result;
}
```

نمونه کد (۱)

۳- تمرین دوم: ضرب دو ماتریس ۱۰۰۰*۱۰۰۰

یکی از برنامه هایی که به صورت موازی می‌توان پیاده سازی کرد محاسبه ضرب دو ماتریس است. در تمرین‌های گذشته به تعداد از این مثال استفاده کردیم که مثال بارز آن استفاده از کتابخانه OpenMP برای تسریع بود. در این تمرین فضای مورد نظر یک ضرب بین دو ماتریس می‌باشد که اندازه ماتریس‌ها مقدار نسبتا بزرگی است. با توجه به وابسته نبودن ضرب ماتریس به صورت ستون به ستون می‌توانیم از کتابخانه POSIX برای ضرب این دو ماتریس استفاده کرد. برای انجام این ضرب از سه حلقه for تو در تو استفاده شده است. جهت دسترسی بهتر به حافظه، در ابتدا ماتریس دوم را ترانهاده کرده و سپس محاسبات را انجام می‌دهیم.

این برنامه را پنج بار اجرا می کنیم. در شکل ۳ نتیجه یکی از اجراها را مشاهده می‌کنیم. در جدول ۳ نیز زمان‌های پنج بار اجرای این برنامه را مشاهده کنید.

```
C:\Users\Shahin\Desktop\HPC\Project1\x64\Release\Project1.exe
Execution time of Serial code: 1.26206 seconds
31044313088.000000,
Execution time of POSIX code: 0.305779 seconds
31044313088.000000,
Speedup from Serial to POSIX = 4.13
```

شکل (۳): نتایج برنامه ضرب دو ماتریس

جدول (۳): زمان اجرای برنامه ضرب دو ماتریس

زمان اجرای سریال	زمان اجرای موازی	میزان بهبود یافته (n برابر)	میانگین بهبود یافته
۱,۲۶۲	۰,۳۰۵	۴,۱۳	۳,۱۷
۱,۳۲۹	۰,۵۸۱	۲,۲۹	
۱,۳۲۱	۰,۴۳۷	۳,۰۲	
۱,۳۲۲	۰,۴۳۱	۳,۰۶	
۱,۳۵۳	۰,۴۰۲	۳,۳۶	

در نمونه کد ۲ می‌توانید بخش‌های مربوط به موازی سازی برنامه را مشاهده نمایید.

```
//Parallel
void *task(void *arg)
{
    int numtread = numtread_g++;
    for (int i = numtread * NUMBER_MAT / NUM_THREADS; i < (numtread + 1
        for (int j = 0; j<NUMBER_MAT; j++) {
            c[i][j] = 0;
            for (int k = 0; k < NUMBER_MAT; k++)
                c[i][j] += a[i][k] * b[j][k];
        }
    }
    return 0;
```

نمونه کد (۲): مربوط به قسمت موازی سازی برنامه Sobel

۴- تمرین سوم: N وزیر

مسئله چند وزیر یک معمای شطرنجی و ریاضیاتی است که بر اساس آن باید n وزیر شطرنج در یک صفحه n×n شطرنج به‌گونه‌ای قرار داده شوند که هیچ‌یک زیر ضرب دیگری نباشند. با توجه به اینکه وزیر به‌صورت افقی، عمودی و اُریب حرکت می‌کند، باید هر وزیر را در طول، عرض و قطر متفاوتی قرار داد.

اولین و مشهورترین شکل این مسئله معمای هشت وزیر است که برای حل آن باید ۸ وزیر را در یک صفحهٔ معمولی

(۸×۸) شطرنج قرار داد. این مسئله ۹۲ جواب دارد که ۱۲ جواب آن منحصر به‌فرد است یعنی بقیه جواب‌ها از تقارن جواب‌های اصلی به‌دست می‌آید.

مسئله n وزیر در صورتی جواب دارد که n مساوی ۱ یا بیشتر از ۳ باشد. یعنی مسئله دو وزیر و سه وزیر راه حلی ندارند. این مسئله در سال ۱۸۴۸ توسط شطرنج بازی به نام Max Bezzel عنوان شد و ریاضی دانان بسیاری ازجمله Gauss و Georg Cantor بر روی این مسئله کار کرده و در نهایت آن را به n وزیر تعمیم دادند. اولین راه حل توسط Franz Nauck در سال ۱۸۵۰ ارائه شد که به همان مسئله n وزیر تعمیم داده شد. پس از آن Gunther راه حلی با استفاده از دترمینان ارائه داد که J.W.L. Glaisher آن را کامل نمود. در سال ۱۹۷۹، Edsger Dijkstra Nauck این مسئله را با استفاده از الگوریتم عقب‌گرد حل کرد. هشت وزیر را می‌توان با الگوریتم‌های مختلفی مانند تپه نوردی و روش ارضای محدودیت(csp) حل کرد. هدف از مسئله n وزیر، چیدن n مهره وزیر در یک صفحه شطرنج(n*n) است، به‌طوری‌که هیچ دو وزیری یکدیگر را گارد ندهند، یعنی هیچ دو مهره‌ای نباید در یک سطر، ستون یا قطر یکسان باشند. وزیر در خانه‌های شطرنج به صورت عرضی، طولی و قطری می‌تواند حرکت کند. مسئله n وزیر از جمله مسائل NP در هوش مصنوعی است که روش‌های جستجوی معمولی قادر به حل آن‌ها نخواهد بود. در ابتدا این تمرین این برنامه به صورت سری پیاده سازی شده است. حال هدف از این تمرین تغییر و پیاده سازی موازی این قطعه کد با استفاده از دستورات کتابخانه POSIX می‌باشد.

جدول (۴): زمان اجرای برنامه N وزیر با ۲ ریسمان

تعداد وزیر	زمان اجرای سریال	زمان اجرای موازی	میزان بهبود یافته (n برابر)
۴	۰,۰۰۱	۰,۱۷۱	۰,۰۱
۸	۰,۳۲۱	۰,۳۹۳	۰,۸۲
۱۲	۱۵۳	۱۳۹	۱,۱

جدول (۵): زمان اجرای برنامه N وزیر با ۴ ریسمان

تعداد وزیر	زمان اجرای سریال	زمان اجرای موازی	میزان بهبود یافته (n برابر)
۴	۰,۰۰۱	۰,۲۱۸	۰,۰۱
۸	۰,۳۰۱	۰,۴۲۱	۰,۷۲
۱۲	۱۵۷	۱۴۲	۱,۱۱

جدول (۶): زمان اجرای برنامه N وزیر با ۸ ریسمان

تعداد وزیر	زمان اجرای سریال	زمان اجرای موازی	میزان بهبود یافته (n برابر)
۴	۰,۰۰۱	۰,۴۱۸	۰,۰۱
۸	۰,۲۵۴	۰,۴۵۳	۰,۵۶
۱۲	۱۵۵	۱۳۹	۱,۱۱

مراجع

[۱] مباحث مربوط به درس محاسبات کارایی بالا

ضمایم

کدهای مربوط به هر بخش در پوشه خود موجود می‌باشند