CA5

ثمین مهدی زاده-حانیه ناصری

سوال ۱)

پیاده سازی **موازی** :

ابتدا پیاده سازی **موازی** با استفاده از posix را شرح می دهیم.

برای پیاده سازی موازی عملیات مربوط به یافتن عدد ماکسیمم و اندیس آن، دو متغیر global، یکی با نام max_value_parallel برای یافتن بزرگ ترین عدد و دیگری max_index_parallel برای نگه داری اندیس آن تعریف می کنیم.

پیاده سازی را می خواهیم به گونه ای انجام دهیم که آرایه را به تعداد نخ ها تقسیم کنیم و هر کدام را مسئول محاسبه ماکسیمم و اندیس آن در قسمت مربوطه اش قرار می دهیم. هر نخ بعد از اتمام تسک خود، باید max_value_parallel و max_index_parallel را با مقادیر محاسبه شده خود آپدیت کند که اینجا برای جلوگیری از ناسازگاری های read/write همزمان، یک mutex داریم که قسمت مربوط به آپدیت کردن max_value_parallel و max_index_parallel را بحرانی در نظر گرفته به گونه ای که در هر لحظه، فقط یک نخ می تواند در آن جا حضور داشته باشد. حال به توضیح جزیی تر کد های زده شده می پردازیم.

ابتدا یک آرایه به تعداد کل نخ ها از pthread_t تعریف می کنیم. سپس آرایه ای جهت مقدار دهی ورودی های تابع محاسبه ماکسییم برای هر نخ تعریف می کنیم که از جنس structure با نام in_param_t می باشد. این structure شامل ۳ فیلد tid ، start_index و end_index می باشد که به ترتیب بیانگر شماره نخ، اندیس شروع بازه متعلق به آن و اندیس پایان بازه است. این آرایه را با مقادیر مناسب برای هر نخ مقدار دهی می کنیم و بعد از آن هم، بر روی متغیر mutex_lock، pthread_mutex_init را صدا می زنیم.

در نهایت، به ازای هر نخ، pthread_create را با پارامتر های هندلر مربوط به آن نخ، نام تابع اجرا شونده توسط نخ و پارامتر های ورودی برای آن نخ (از جنس in_param_t و کست شده به *void) صدا می زنیم تا نخ ها موازی تسک خود را انجام دهند.

در داخل تابع find_local_max، هر نخ متغیر های لوکال و شخصی local_max و local_index خودش را دارد. متغیر arg برای این تابع را که از جنس *void است، به * in_param_t کست کرده (با نام inp) و از داخل آن اندیس شروع و پایان بازه نخ را یافته (inp->start_index و inp->end_index) و بر روی آن حلقه میزنیم تا max_local و max_index آپدیت شوند.

در پایان حلقه for، روی قفل mutex_lock صبر کرده و در صورتی که نخ دیگری داخل ناحیه بحرانی مربوط به آپدیت ماکسیمم نهایی نباشد، وارد آن شده و ماکسیمم نهایی و اندیس آن را با لوکال خود آپدیت می کنیم.

نخ اصلی با pthread_join بر روی همه نخ های ایجاد شده منتظر اتمام آنها ست و بعد اتمام کار همه نخ ها، ماکسیمم نهایی و اندیس نهایی را چاپ می کنیم (max_local و max_index).

پیاده سازی **سریال**:

در نسخه **سریال**، کل حلقه ماکسیمم توسط همان نخ اصلی اجرا کننده برنامه با یک حلقه for از ابتدا تا انتها بررسی و اجرا می شود و طبق روال عادی ماکسیمم و ایندکس آن که max_value_serial و max_index_serial نام دارند، محاسبه می شوند.

تصویر اجرای برنامه را در زیر مشاهده می نمایید:

```
mac@macs-MacBook-Pro desktop % g++ Q1.cpp -o Q1 -lpthread
mac@macs-MacBook-Pro desktop % ./Q1
810196623 - 810196573
parallel
max value: 99.999702 , max index: 797174
serial
max value: 99.999702 , max index: 797174
PARALLEL Execution time is 0 s and 1906 micros
SERIAL Execution time is 0 s and 2967 micros
improvement : 1.556663
mac@macs-MacBook-Pro desktop % ./Q1
810196623 - 810196573
parallel
max value: 99.999817 , max index: 288310
serial
max value: 99.999817 , max index: 288310
PARALLEL Execution time is 0 s and 1568 micros
SERIAL Execution time is 0 s and 3167 micros
improvement : 2.019770
mac@macs-MacBook-Pro desktop % ./Q1
810196623 - 810196573
parallel
max value: 99.999840 , max index: 964938
serial
max value: 99.999840 , max index: 964938
PARALLEL Execution time is 0 s and 1780 micros
SERIAL Execution time is 0 s and 3000 micros
improvement : 1.685393
mac@macs-MacBook-Pro desktop % ./Q1
810196623 - 810196573
parallel
max value: 99.999863 , max index: 390971
serial
max value: 99.999863 , max index: 390971
PARALLEL Execution time is 0 s and 1573 micros
SERIAL Execution time is 0 s and 2918 micros
improvement : 1.855054
mac@macs-MacBook-Pro desktop % ./Q1
810196623 - 810196573
parallel
max value: 99.999962 , max index: 538792
serial
max value: 99.999962 , max index: 538792
PARALLEL Execution time is 0 s and 1679 micros
SERIAL Execution time is 0 s and 3266 micros
improvement : 1.945205
mac@macs-MacBook-Pro desktop % ./Q1
810196623 - 810196573
parallel
max value: 99.999825 , max index: 219895
serial
max value: 99.999825 , max index: 219895
PARALLEL Execution time is 0 s and 1877 micros
SERIAL Execution time is 0 s and 3068 micros
improvement : 1.634523
mac@macs-MacBook-Pro desktop %
```

برای محاسبه و مقایسه زمان های اجرای دو پیاده سازی، از gettimeofday استفاده کردیم.

سوال دوم)

ابتدا الگوریتم Quick Sort را مختصرا توضیح می دهیم. در مرتب سازی quick sort، از بین اعضای آرایه، یک عضو را به عنوان pivot در نظر میگیریم. بدین صورت که از میان اعضای آرایه، یک عضو را در نظر گرفته و عضو های کوچکتر از آن را سمت چپ آن و عضو های بزرگتر را در سمت راست آن قرار می دهیم (در اینجا pivot را برابر با عنصر آخر ارایه (اندیس آخر) در نظر می گیریم). حال برای هر کدام از دو تکه ایجاد شده (تکه آرایه قبل pivot و تکه آرایه بعد آن) عمل فوق را دوباره انجام می دهیم. اگر آرایه اولیه را به عنوان ریشه درخت در نظر بگیریم، در هر مرحله بر حسب pivot دو بچه تولید می شوند و با ادامه دادن آن در هر شاخه در نهایت به آرایه مرتب شده به صورت صعودی میرسیم.

برای پیاده سازی این الگوریتم در دو حالت سری و موازی تابع quicksort زده شده است.از آن جا که ورودی ها و خروجی تابعی که توسط thread ایجاد می شود باید از نوع *void باشد یک استراکت(in_param_t)تعریف شده که به کمک آن ورودی ها به تابع داده شود این استراکت شاملlow,high(بازه ی ابتدایی و انتهایی آرایه)وtype است که در صورتی که type برابر یک باشد عملیات بر روی آرایه سریال و درغیر این صورت بر روی آرایه موازی انجام می شود(از آن جا که پس از انجام عملیات آرایه مرتب شده و ارسال آرایه با طول زیاد در تابع زمان بر است جهت مقایسه ی زمان اجرا دو آرایه کاملا شبیه به هم در ابتدا ایجاد شده است)

برای quicksort از یک آرایه به نام استک استفاده شده است و همچنین اشاره گری به سر استک نگه داشته می شود.در ابتدا بازه های اولیه ی ابتدا و انتهای آرایه به استک داده می شود.در هر مرحله دو عنصر به عنوان low و high از استک برداشته می شوند و پارتیشن بر روی آن ها انجام می شود.پس از به وجود آوردن partition ابتدا و انتهای بازه های به وجود آمده دوباره به استک اضافه می شود تا با برداشتن آن ها از استک دوباره عمل partition که منجر به مرتب سازی آرایه می شود اتفاق بیافتد. این کار آنقدر ادامه می یابد تا استک خالی شده و دیگر مرزی باقی نمانده باشد.در این حالت همه ی بازه های ابتدایی و انتهایی مرتب شده هستند.در ادامه چگونگی پیاده سازی این سورت در دو قسمت موازی و سریال آمده است.

پیاده سازی **سریال:**

In_param[8] برای ورودی سریال درنظر گرفته شده است بنابراین0 = type و مقادیر low,high به ترتیب برابر DATA_SIZE-1 و 0 قرار می گیرند.سپس این متغیر به عنوان ورودی به تابع quicksort داده می شود تا روی آن سورت داده ها انجام شود.

پیاده سازی **موازی:**

در اینجا ابتدا به جای آنکه از ابتدا تابع quicksort را بر روی کل آرایه صدا بزنیم، ابتدا ۷ بار partition روی آن صدا میزنیم تا در هر بار یک pivot انتخاب شود و عناصر نسبت به آن در قبل یا بعد آن قرار بگیرند. پس آرایه ۸ قسمت می شود که در هر قسمت، عناصر از pivot قبل خود بزرگتر و از pivot بعد خود کوچکتر هستند (فقط توجه کنید که ممکن است در اجرای یک partition اندیس pivot بعد از تغییر ساختار آرایه جایی قرار بگیرد که تکه ای بعد آن یا قبل آن نباشد و بنابر این ۴ partition بعد آن تحت تاثیر قرار می گیرند که برای هندل کردن این موضوع، ابتدا اندیس تمام هفت pivot را ۱- در نظر گرفتیم و بعد اگر برای partition مقدار اندیس پایین کمتر از اندیس بالای تکه بود و pivot قبل یا بعد ۱- نبودند، partition را بر روی آن تکه صدا زده و اندیسpivot را آپدیت می کنیم. حال هر یک از ۸ قسمت را به ۸ نخ می دهیم تا به موازات هم quicksort را روی آن اجرا کنند.در انتها نیز منتظر میمانیم تا کار تمام نخ ها به اتمام برسد.

تست:

برای تست درستی برنامه طول آرایه را ۱۰ در نظر گرفته و نتایج اجرای سریال و موازی را مشاهده می کنیم:

```
mac@macs-MacBook-Pro desktop % g++ Q2.cpp -o Q2 -lpthread
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
orginal array:
52.505993 68.253799 41.644417 17.732983 38.244457 74.622482 80.097290 95.100250 49.882179 69.787308
PARALLEL Execution time is 0 s and 466 micros
SERIAL Execution time is 0 s and 21 micros
improvement : 0.045064
serial sort:
17.732983 38.244457 41.644417 49.882179 52.505993 68.253799 69.787308 74.622482 80.097290 95.100250
parallel sort:
17.732983 38.244457 41.644417 49.882179 52.505993 68.253799 69.787308 74.622482 80.097290 95.100250
mac@macs-MacBook-Pro desktop %
```

مشاهده می شود که آرایه در هر دو روش به درستی سورت شده است.ولی چون تعداد عناصر کم است improvement نداریم.

اجرای برنامه با تعداد داده های بالا:

```
mac@macs-MacBook-Pro desktop % g++ Q2.cpp -o Q2 -lpthread
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 0 s and 146043 micros
SERIAL Execution time is 1 s and 213918 micros
improvement : 1.471608
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 0 s and 107639 micros
SERIAL Execution time is 1 s and 205974 micros
improvement : 1.922853
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 0 s and 160435 micros
SERIAL Execution time is 0 s and 213441 micros
improvement : 1.330389
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 1 s and 108240 micros
SERIAL Execution time is 0 s and 209993 micros
improvement : 1.922309
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 0 s and 153795 micros
SERIAL Execution time is 0 s and 216376 micros
improvement : 1.406912
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 0 s and 127028 micros
SERIAL Execution time is 0 s and 206246 micros
improvement : 1.623626
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 0 s and 110450 micros
SERIAL Execution time is 1 s and 210784 micros
improvement : 1.917465
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 0 s and 185879 micros
SERIAL Execution time is 0 s and 210783 micros
improvement : 1.133980
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 0 s and 115620 micros
SERIAL Execution time is 1 s and 210024 micros
improvement : 1.825151
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 0 s and 118846 micros
SERIAL Execution time is 0 s and 210948 micros
improvement : 1.774969
mac@macs-MacBook-Pro desktop % ./Q2
810196623 - 810196573
PARALLEL Execution time is 0 s and 119115 micros
SERIAL Execution time is 0 s and 210341 micros
improvement : 1.765865
mac@macs-MacBook-Pro desktop %
```