به نام خدا

برنامه‌نویسی چندهسته‌ای

دستور کار آزمایشگاه ۴

هدف از این آزمایش، انجام عملیات Prefix sum بر روی یک آرایه است. این عملیات به صورت زیر تعریف می‌شود:

$$y[i] = \sum_{j=0}^{i} x[j]$$

به عبارت دیگر، هر درایه در آرایه خروجی، جمع همه درایه‌های قبل از خود در آرایه ورودی است. به عنوان مثال:

آرایه ورودی:

آرایه خروجی:

x:	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
y:	۱	۳	۶	۱۰	۱۵	۲۱	۲۸	۳۶	۴۵	۵۵

این عملیات کاربرد فراوانی در حوزه‌های مختلف دارد. به عنوان مثال فرض کنید آرایه x همه تراکنش‌های مالی یک حساب (میزان کاهش یا افزایش حساب) است و آرایه y مقدار موجودی حساب تا پایان هر تراکنش. به عنوان مثال دیگر، فرض کنید آرایه x مقادیر تابع احتمالی PDF و آرایه y مقادیر تابع احتمالی تجمعی CDF است.

کد سریال این الگوریتم در فایل سرور به نام lab_4_serial.c وجود دارد.

در این آزمایش، دو روش موازی سازی الگوریتم Prefix sum معرفی می‌شود[1].

۱- روش اول:

ابتدا آرایه x را بین نخ‌ها به صورت static تقسیم کنید (هر نخ 1/n آرایه را پردازش می‌کند). هر نخ عملیات prefix sum را به صورت مستقل بر روی زیرآرایه خود انجام می‌دهد. به عنوان مثال با دو نخ:

x:	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
y:	۱	۳	۶	۱۰	۱۵	۶	۱۳	۲۱	۳۰	۴۰

[1] https://scs.senecac.on.ca/~gpu621/pages/content/omp_4.html

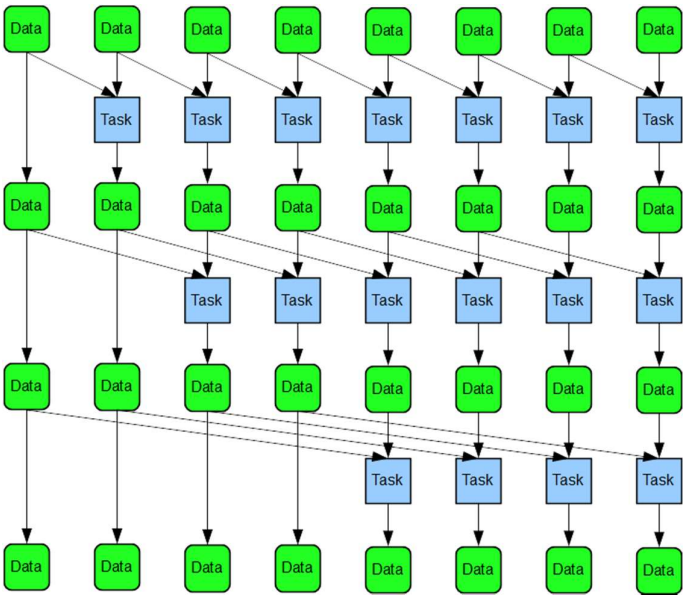
سپس می‌بایست مقدار خانه آخر هر زیر آرایه را با تمامی خانه‌های زیرآرایه‌های بعد از خود جمع کنیم. در مثال بالا، مقدار خانه آخر زیرآرایه اول (۱۵) باید با همه خانه‌های زیرآرایه دوم جمع شود. چرا؟

دقت کنید این کار باید برای همه زیرآرایه‌ها انجام شود. تصور کنید اگر تعداد نخ‌ها چهار بود چه می‌شد؟

کد موازی این الگوریتم را بنویسید و زمان اجرای آن را با حالت سریال مقایسه کنید.

۲- روش دوم:

یک الگوریتم برای محاسبه prefix scan الگوریتمی است به نام Hillis and Steele که در سال ۱۹۸۶ معرفی شده است. شکل زیر الگوی محاسبات را نشان می‌دهد:



در این شکل، آرایه ورودی دارای ۸ المان و در بالا نشان داده شده و آرایه خروجی در پایین محاسبه شده است. هر مربع task یک جمع است.

کد موازی این الگوریتم را بنویسید و زمان اجرای آن را با حالت سریال مقایسه کنید.

الگوریتم دوم با اینکه کار موازی بیشتری تولید می‌کند ولی کندتر از الگوریتم اول و حتی الگوریتم سریال است. چرا؟

توضیح دهید الگوریتم دوم در چه حالتی می‌تواند نسبت به الگوریتم اول مزیت داشته باشد (راهنمایی: این الگوریتم در GPU بسیار پرکاربرد است).