Yasaman Mirmohammad
Zahra Yousefi

به نام خدا

برنامه‌نویسی چندهسته‌ای

گزارش کار آزمایشگاه ۲

مقدمه

در این آزمایش شما باید یک برنامه سریال جمع دو ماتریس را با تنظیمات مناسب در محیط Visual Studio کامپایل و اجرا کنید. سپس به کمک رهنمود[1]های OpenMP برنامه را موازی و از صحت عملکرد آن اطمینان حاصل کنید. در نهایت تسریع به‌دست‌آمده محاسبه می‌شود.

آزمایش

❖ **مرحله اول: ساخت پروژه در ویژوال استودیو، تنظیم پروژه، کامپایل و اجرای کد**

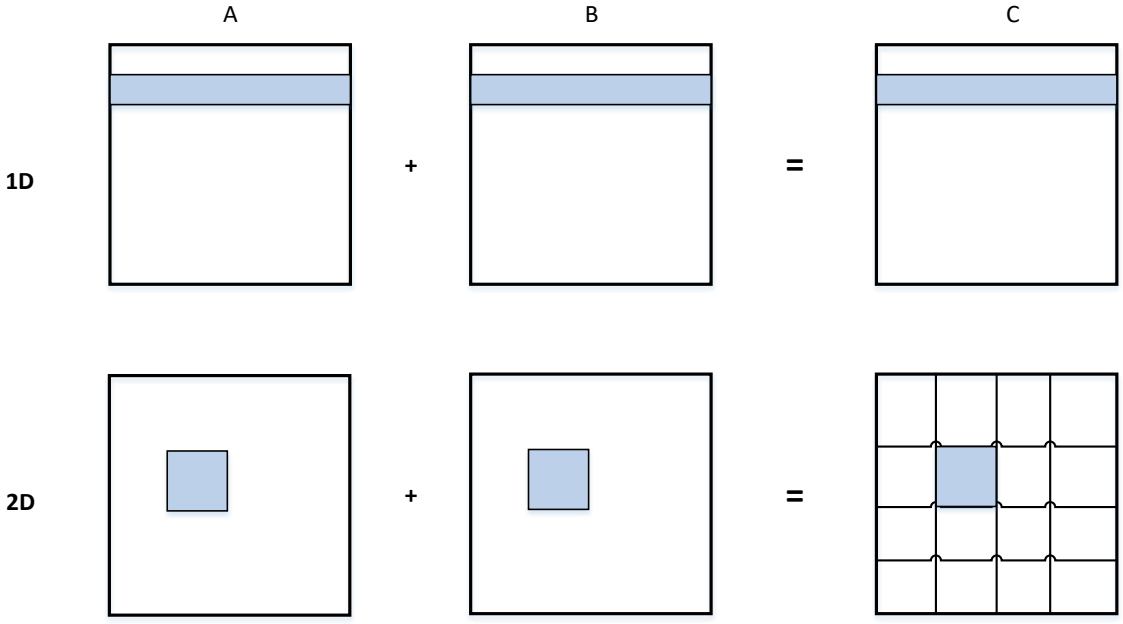
- کد برنامه درون فایل zip که در اختیارتان قرار گرفته موجود است. نام فایل matadd.cpp است.
- به حالت کامپایل (Debug/Release) توجه کنید.

❖ **مرحله دوم: فعال‌سازی OpenMP و موازی‌سازی برنامه**

برنامه را به کمک OpenMP به سه صورت زیر موازی کنید و خروجی آن را با خروجی کد سریال مقایسه کنید:

۱. یک بعدی سطری
۲. یک بعدی ستونی
۳. دوبعدی

قسمت حاشور زده شده حجم کاری است که یک نخ انجام می‌دهد (شکل اول روش یک بعدی سطری را نشان می‌دهد). توجه داشته باشید که صرفاً موازی‌سازی عمل جمع مد نظر است. برای موازی سازی دوبعدی می‌توانید از nested parallelism استفاده کنید. توجه داشته باشید باید این ویژگی فعال شود.



[1] Directive

❖ **مرحله سوم: اندازه‌گیری**

برای سادگی ماتریس‌های A و B را مربعی فرض کنید و پس از پیاده‌سازی، زمان عمل جمع را با تابع مناسب اندازه گرفته و برای هر سه روش موازی‌سازی جدول ذیل را پر کرده و گزارش کنید. تکرار هر اجرا و میانگین گرفتن از زمان‌های اجرا به افزایش دقت اندازه‌گیری کمک می‌کند. ابعاد هر ماتریس ورودی را به گونه‌ای بگیرید که حجم آن برابر مقدار ورودی خواسته شده باشد. هر ***int*** را چهار بایت فرض کنید.

گزارش:

ما از تابع **double omp_get_wtime**(void); استفاده کردیم.

یک زمان مشخص برای لاگ گرفتن از زمان سپری شده ی اجرای تابع ADD در نظر میگیریم. تا ببینیم اجرای این قسمت از برنامه چقدر به طول انجامیده است.

نتایج روش اول

تعداد نخ‌ها	اندازه هر ماتریس ورودی				تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1	0.006	0.01	1.02	0.4	-
4	0.004	0.004	0.2	0.2	
8	0.014	0.01	0.22	0.2	
16	0.03	0.02	0.20	0.2	

نتایج روش دوم

تعداد نخ‌ها	اندازه هر ماتریس ورودی				تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1	1.133	0.01	16	16	-
4	0.004	0.005	5.6	6.6	
8	0.01	0.01	5.3	5.5	
16	0.02	0.01	4	4.5	

نتایج روش سوم

تعداد نخ‌ها	اندازه هر ماتریس ورودی				تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1	0.01	0.007	1.1	0.4	-
4	0.02	0.02	0.3	0.2	
8	0.09	0.05	0.5	0.3	
16	0.3	0.1	1.04	0.7	

Yasaman Mirmohammad
Zahra Yousefi

❖ **گزارش**

پس از اتمام آزمایش به سوال‌های زیر پاسخ دهید:

۱. **کدام یک از دو روش تجزیه یک بعدی کندتر اجرا می‌شود؟ چرا؟**

ستونی. به این علت که پیمایش ماتریس مربعی ما به صورت دو FOR تو در تو است و اگر ابتدا سطر ها پیمایش شوند در تجزیه ی یک بعدی رویکرد ما سریعتر خواهد شد.

پس رویکرد ستونی، کندتر است.

۲. **افزایش اندازه ماتریس ورودی چه تاثیری بر میزان تسریع دارد؟**

افزایش ابعاد ماتریس تاثیر مستقیمی بر میزان تسریع اجرا در اصر موازی سازی دارد. به عبارتی تاثیر موازی سازی عمل جمع، در ابعاد بالا بهتر مشخص میشود.

۳. **بیشترین میزان تسریع برای هر روش مربوط به چه تعداد نخ است؟ این تعداد چه ارتباطی با ساختار پردازنده شما دارد؟**

مربوط به ۴ نخ است. ساختار پردازنده ی من ۴ نخی است و انتظار میرفت نتیجه همین باشد.

۴. **از نظر شما روش ایستا برای تقسیم کار بین نخ‌ها در این برنامه مناسب‌تر است یا روش پویا؟ چرا؟**

در حالت کلی می توان گفت روش ایستا برای مواقعی کاربرد دارد که workload الگوریتم ما معقول است و میزان کار یکسانی بر دوش هر نخ می افتد. مثلا در عملیات کانولوشن و....

اما هنگامی که معقول نباشد، از روش دینامیک استفاده میشود تا به هر نخ میزان کار متناسبی تخصیص داده شود.(وزن)

۵. **به طور کلی تجزیه دو بعدی و یک بعدی چه مزایایی نسبت به هم دارند؟ به طور خاص در این برنامه کدام روش بهتر عمل می‌کند؟**

مزیت اصلی تجزیه ی دو بعدی این است که ابعاد داده را بیشتر کاهش میدهد. این ویژگی در خصوص ماتریس هاس تنک که مقادیر زیادی صفر دارند بهتر جواب میدهد. وقتی ماتریس مربعی نباشد تجزیه ی دو بعدی بهتر است.

اما در برنامه ی ما تجزیه ی یک بعدی نتیجه ی بهتری میدهد.

بخش امتیازی

پاسخ این قسمت را می توانید پس از آزمایشگاه در قسمتی که برای ارسال کدها و نیز گزارشکارها باز شده است ارسال نمایید.

علاوه بر روش های بالا روش زیر نیز می تواند به ما در بهینگی عملکرد برنامه موازی داده شده کمک کند.

۴. تجمیع حلقه‌های for

یک مشکل برای روش یک بعدی سطری این است که تمام عناصر یک سطر توسط یک نخ پردازش می شوند. می‌دانیم ماتریس های دوبعدی در حافظه به صورت یک آرایه یک بعدی ذخیره می شوند بنابراین می توانیم با تجمیع دو حلقه for موجود برای پیمایش آرایه دو بعدی به یک حلقه عمل جمع ماتریس ها را به صورت بهتر و نزدیک‌تری به نحوه‌ی پیاده سازی آن توسط سخت افزار انجام دهیم. هم چنین نسبت به روش دو بعدی سربار کمتری را نیز به سیستم تحمیل کنیم. این کار را به صورت زیر می توانیم انجام دهیم:

```
void add_collapse(DataSet dataSet, int chunkSize) {
#pragma omp parallel for schedule(static, chunkSize)
	for (int ij = 0; ij < dataSet.n * dataSet.m; ij++) {
		dataSet.C[ij] = dataSet.A[ij] + dataSet.B[ij];
	}
}
```

در این روش به اصطلاح دو حلقه for موجود را در هم collapse کردیم. دقت کنید که پس از collapse کردن همچنان می توانیم i را از طریق انجام عملیات ij / dataset.m و j را از طریق ij % dataSet.m به دست آوریم.

این عملیات در openMP های ورژن ۳ به بالا اضافه شد تا بتوان بدون تغییر ظاهر کد نیز از روش collapse کردن استفاده نمود:

```
void add_collapse(DataSet dataSet, int chunkSize) {
#pragma omp parallel for collapse(2) schedule(static, chunkSize)
	for (int i = 0; i < dataSet.n; i++) {
		for (int j = 0; j < dataSet.m; j++) {
			dataSet.C[i * dataSet.m + j] =
				dataSet.A[i * dataSet.m + j] + dataSet.B[i * dataSet.m + j];
		}
	}
}
```

Yasaman Mirmohammad
Zahra Yousefi

با پیاده سازی روش بالا جدول زیر را تکمیل کنید و زمان‌ها را با سایر روش‌ها مقایسه نمایید.

تعداد نخ‌ها	اندازه هر ماتریس ورودی				تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1					-
4					
8					
16					

میزان تسریع این روش را نسبت به سایر روش‌ها تحلیل نمایید.