تمرین کامپیوتری دوم درس محاسبات با کارایی بالا دکتر صفری

شاهین منتظری ۹۱۰۱۹۷۳۹۰

محمد هاشمی ۹۱۰۱۹۷۴۲۳

چکیده - در این تمرین به طور کلی به بررسی و اجرای برنامه های کامپیوتری موازی در کتابخانه OpenMP می پردازیم. با مقایسه ی برنامه های مختلف چه به صورت سریال و چه به صورت موازی با تعداد ریسه های متعدد در می یابیم که از نظر کارآیی تا چه مقدار بهبود حاصل می شود. در بخش اول تمرین، الگوریتم های Sobel و Absolute Difference که به ترتیب به منظور تشخیص لبه های تصویر و تشخیص حرکت اجسام به کار برده می شوند، با استفاده از دستورات OpenMPموازی پیاده می شود و از نظر زمانی با برنامه های کامپیوتری مربوطه در تمرین شماره ۱ که به صورت SIMD موازی شده بودند، مقایسه می گردند تا میزان تسریع مشخص شود. با توجه به نتایج به دست آمده این میزان تسریع برنامه Sobel برابر ... و میزان تسریع برنامه Absolute Difference برابر ... می باشد. در بخش های ۳ تا ۵ این تمرین نیز برنامه های دیگری از جمله ضرب ماتریس و شبکه کانولوشنی سنتی به دو صورت سریال و موازی OpenMP قرار دارند که لازم است با اعمال تغییراتی اجرا شوند تا نتایج حاصل از میزان تسریع نمایان شود. این جداول و نمودارهای این نتایج در بخش های مربوطه ارائه می گردند.

کلید واژه – OpenMP - C + + - Parallel Computing - Image Processing - Sobel Edge Detection Operator – Convolutional Neural Networks

۱- مقدمه

OpenMP (Open Multi - Processing) یک رابط برنامه کاربردی Application API Programming Interface است که سیستم عمال های مختلفی از جمله Solaris , AIX , Linux , MacOSx و Windows از آن پشتیبانی می کنند.OpenMP یک واسط انعطاف پذیر ساده را در اختیار برنامه نویسان قرار می دهد تا کاربرد های موازی برای کامپیوترها فراهم نماید. OpenMP یک پیاده سازی از چند ریسه ای (multi - thread) است، روشی از موازی سازی که به وسیله آن master thread (مجموعه ای از دستورالعمل ها که به صورت پی در پی اجرا می شود) به تعداد مشخصی slave thread شکسته می شود و وظیفه ای را بین آنها تقسیم می نماید. به هر ریسه یک شناسه تعلق می گیرد. شناسه ریسه عددی صحیح است و ریسه اصلی شناسه صفر دارد. بعد از اجرای برنامه ی موازی ریسه ها دوباره متصل شده تا به ریسه اصلی برسند و سپس برنامه ادامه می باید تا به انتها برسد. به طور پیشفرض هر ریسه بخش موازش شده ای را به صورت مستقل اجرا می کند. تقسیم وظایف بین ریسه ها توسط مولفه های تقسیم کار یا Work - Sharing Constructs انجام می شود. به طور کلی هم انجام وظایف به طور موازی و هم تسهیم داده با استفاده از OpenMP با همین روش به دست می آید. تخصیص ریسه به پردازنده بر اساس فاکتورهایی مانند Usage و Machine Load انجام می شود. تعداد ریسه ها می تواند به وسیله محیط زمان اجرا بر اساس متغیرهای محیطی و یا در کد با استفاده از Directive یا تابع خاص تخصیص داده شود. توابع OpenMP در فایل h.omp در زبان c /c++ موجود می باشد.OpenMP بر اساس فضای اشتراکی بنا شده است و مرتبط با ریسه های همزمان می باشد که نحوه می موازی سازی را مدیریت می کند. به ریسه اصلی فعال هم گفته می شود که صرفا تنها ریسه زنده است. این ریسه از جایی که ریسه های همزمان منشعب می شوند تا زمانی که به هم می پیوندند، وجود دارد. به CPU که حافظه اشتراکی دارد چند پردازنده متقارن یا SMP نامیده می شود. هر ریسه روی پردازنده خودش اجرا می شود اما حالتی که هر پردازنده یا هسته بتواند بیش از یک ریسه را به طور همزمان اجرا و مدیریت می کند، -

Hyper Threading نامیده می شود.
ارتباط بین ریسه ها به صورت ضمنی است یعنی از متغیرهای استفاده می کند که به مکان حافظه اشتراکی اشاره می نماید که این نحوه ارتباط درست در مقابل MPI قرار دارد.MPI از پیام های صریح برای ارتباط بین هر ریسه استفاده می نماید.
پردازش تصویر یا Image Processing ، امروزه به عنوان یکی از مولفه های اساسی در سیستم های هوشمند و پشتیبان تصمیم است، که غالبا بر روی تصاویر دیجیتال و توسط سیستم های کامپیوتری اعمال می شود. کاربردهای متنوعی که پردازش تصویر در زمینه های مختلف فنی، صنعتی، شهری، پزشکی و علمی دارد، آن را به یک موضوع بسیار فعال در میان زمینه های پژوهشی تبدیل کرده است.
OpenCV: بینایی ماشین متن بازی یک کتابخانه متن باز شامل بیش از صدها الگوریتم بهینه سازی شده به زبان C و C++ برای تحلیل تصویر و ویدیو است، که از زمان معرفی آن در سال ۱۹۹۹، به میزان زیادی از سوی جامعه محققین و توسعه دهندگان بینایی ماشین به عنوان ابزار توسعه پایه پذیرفته شده است OpenCV .در ابتدا در اینتل به منظور توسعه تحقیقات در زمینه بینایی ماشین و ارتقا کاربردهایی که شدیدا از پردازنده استفاده می کنند، توسعه داده شد. مزیت اصلی OpenCV، در سرعت اجرای آن به خصوص در کاربردهای بی درنگ و البته متن باز بودن و رایگان بودن آن است. این مجموعه آموزشی، تلاشی است برای آشنایی هر چه بیشتر جامعه محققین بینایی ماشین با این کتابخانه ارزشمند، که به صورت گام به گام و عملی همراه با مجموعهٔ متنوعی از مثال ها، شما را برای توسعه ی برنامه های کاربردی خود آماده خواهد ساخت.

۲- تمرین اول : پیاده سازی انتگرال برای بدست آوردن عدد پی به شیوه سری و موازی با OpenMP:

برنامه داده شده را با کمک OpenMP به صورت موازی نوشتیم و زمان اجرای برنامه سریال و موازی را بدست آورده و میزان بهبود سرعت را بدست آوردیم.
اگر با تعداد تکرارهایی که در سوال از ما خواسته شده است و با تعداد threadهای ۲ را اجرا کنیم تقریبا Gain ۱,۳۹ برابر می گیریم و اگر با تعداد تکرار های ۱۰,۰۰۰,۰۰۰ برنامه را اجرا کنیم Gain به ۱,۸۷ بهبود می یابد. زیرا هنگام ساخت thread سرباری را به برنامه اعمال می کنیم و این سربار از Gain نهایی ما می کاهد ولی زمانی که از تعداد تکرارها را زیاد می کنیم میزان سربار نسبت به Gain حاصل شده است اجرای موازی برنامه مقدار ناچیزی می شود و Gain نهایی بهبود می یابد. در جدول ۱ نتایج حاصل از ۵ بار اجرای این برنامه با تکرارهای ۱۰۰,۰۰۰ و ۱۰,۰۰۰,۰۰۰ آورده شده است.

جدول ۱: زمان اجرای محاسبه pi

تعداد تکرار حلقه	زمان اجرای سریال	زمان اجرای موازی	Gain	میانگین Gain
۱۰^۵	۰,۴۷۰	۰,۳۲۳	۱,۴۵	۱,۳۹
	۰,۴۷۲	۰,۳۲۲	۱,۴۷	
	۰,۴۸۳	۰,۳۴۸	۱,۳۹	
	۰,۴۶۹	۰,۳۳۳	۱,۴۱	
	۰,۴۷۱	۰,۳۸۶	۱,۲۲	
۱۰^۷	۴,۷۰	۲,۲۴	۱,۹۴	۱,۷۸
	۴,۷۰	۲,۶۴	۱,۷۸	
	۴,۶۲	۲,۵۶	۱,۸۴	
	۴,۷۱	۳,۱۲	۱,۵۱	
	۴,۷۲	۲,۵۴	۱,۸۵	

۳- تمرین دوم : محاسبه اختلاف دو تصویر پیاپی برای تشخیص حرکت اجسام در یک ویدیو و پیاده سازی این روش به صورت سری و موازی و بررسی سرعت بدست آمده از پیاده سازی موازی

یکی از راه های تشخیص یک جسم متحرک در دو فریم متوالی A و B ، محاسبه‌ی B-A است. یعنی با به دست آوردن قدر مطلق تقاضل یا Absolute Difference دو ماتریس که حاوی دو تصویر A و B باشد، می توان حرکت یا جابه جایی یک جسم در فریم را شناسایی نمود. همچنین برای یافتن اختلاف دو تصویر نیز می توان از همین روش استفاده نمود. شرط لازم این است که ابعاد و تعداد پیکسل های دو تصویر با هم برابر باشد.

در تعمیم این روش می توان به الگوریتم Sum of Absolute Difference اشاره نمود. در این الگوریتم، با استفاده از یک ماسک مثلا با ابعاد ۳*۳ بین دو تصویر، می توان شبیه ترین بخش‌ها یا بخش‌های مشترک را استخراج نمود. این بحث خارج از تمرین حاضر می باشد. در تمرین حاضر ما صرفا به پیاده سازی الگوریتم یافتن اختلاف دو تصویر یا همان Absolute Differnce می پردازیم.

۳-۱: برنامه کامپیوتری اختلاف دو تصویر با استفاده از OpenMP

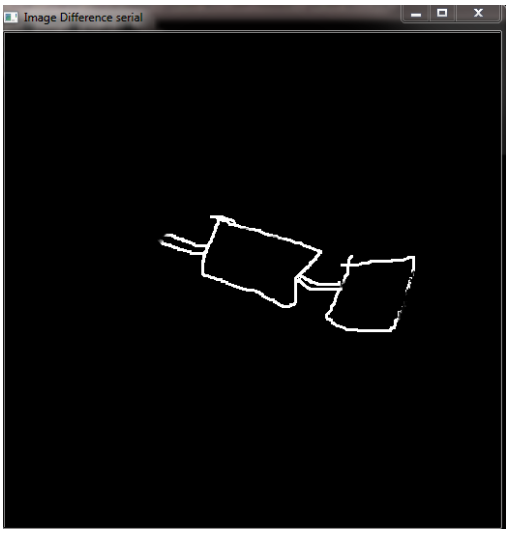
در این بخش کد سریال C++ با استفاده از OpenCV و سپس کد موازی OpenMP همان الگوریتم نوشته شده است. نرم افزار مورد استفاده Visual Studio می باشد. این برنامه نیز مانند برنامه های بخش های قبل به صورت Generie تعریف شده است یعنی کافی است نام دو تصویر مورد نظر را در قسمت Arguments وارد نموده و برنامه را اجرا نماییم. تصویر خروجی با نام دلخواه ذخیره و نمایش داده می شود.

در قسمت برنامه OpenMP به طور کلی از ساختار Parallel for استفاده نمودیم و با آزمون و خطا بر روی موازی سازی حلقه های درونی و بیرونی، تعداد ریسه ها tumrolling حلقه ها و - بهترین حالت ممکن را به منظور به حداقل رساندن زمان اجرای این برنامه به دست آمد.

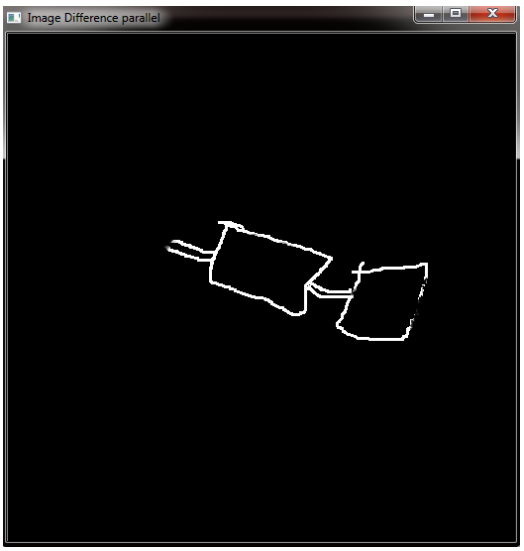
نمونه ای از تصاویر ورودی و خروجی و همچنین زمان اجرای بخش های سریال که توسط دستور های C++ انجام شده و همچنین موازی که توسط OpenMP انجام شده در شکل‌های زیر نشان داده شده است.

الف) تصویر ورودی اول

ب) تصویر ورودی دوم



ج) خروجی حاصل از برنامه سریال

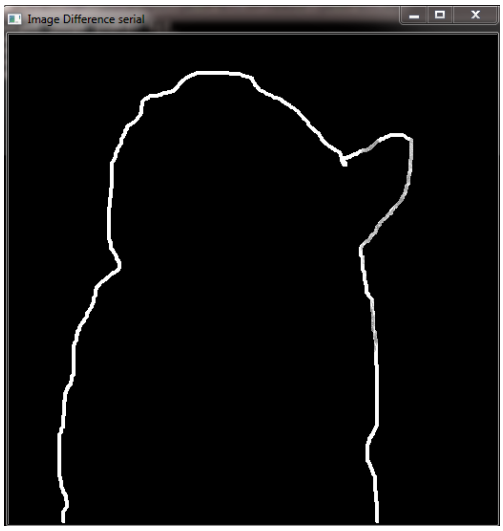


د) خروجی حاصل از برنامه موازی

شکل(۱): نتایج حاصل از شکل اول

الف) تصویر ورودی اول

ب) تصویر ورودی دوم



ج) خروجی حاصل از برنامه سریال

د) خروجی حاصل از برنامه موازی

شکل(۲): نتایج حاصله از شکل دوم

جدول ۲: زمان اجرای اختلاف تصاویر

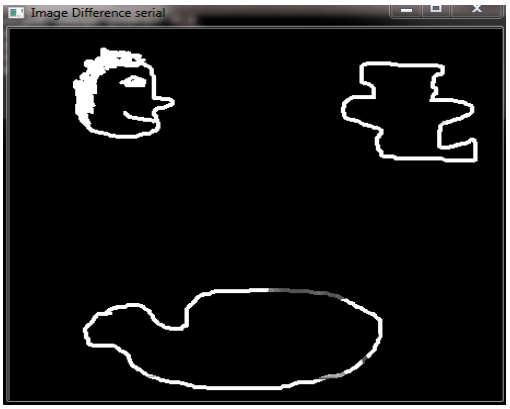
شماره تصویر	زمان اجرای سریال	زمان اجرای موازی	Gain	میانگین Gain
۱	۲,۵۴	۱,۸۲	۱,۳۹	۲,۰۰
	۵,۹۲	۲,۰۲	۲,۹۲	
	۳,۱۲	۱,۶۰	۱,۹۵	
	۱,۳۸	۷,۲۹	۱,۹۰	
	۲,۹۸	۱,۵۹	۱,۸۷	
۲	۲,۹۲	۰,۸۱۵	۳,۵۹	۲,۶۵
	۱,۲۱	۰,۷۱۹	۱,۶۸	
	۲,۸۴	۰,۸۱۳	۳,۵۰	
	۲,۰۶	۰,۷۲۰	۲,۸۶	
	۱,۲۳	۰,۷۵۲	۱,۶۴	
۳	۱,۷۴	۰,۶۵۴	۲,۶۷	۲,۴۸
	۱,۳۴	۰,۶۱۷	۲,۱۷	
	۲,۴۲	۰,۴۷۸	۳,۵۷	
	۱,۵۲	۰,۶۳۵	۲,۴۰	
	۱,۰۱	۰,۶۳۱	۱,۵۹	

همانطور که مشاهده می شود تفاوتی بین خروجی های این برنامه ها مشاهده نمی شود در صورتی که به طور میانگین مدت زمان اجرای دستور های موازی حدود ۲,۵ برابر نسبت به زمان اجرای دستور های سریال کمتر می‌باشد و این یعنی Speed Up حدود ۲,۵ برابر برای OpenMP. البته این مقدار برای تصویر نسبتا کوچک با ابعاد ۵۱۲*۵۱۲ است و برای تصاویر بزرگتر بیشتر نمایان خواهد شد.

کدها و تصاویر مربوط به این بخش در فایل "PartB" می‌باشد.

الف) تصویر ورودی اول

ب) تصویر ورودی دوم



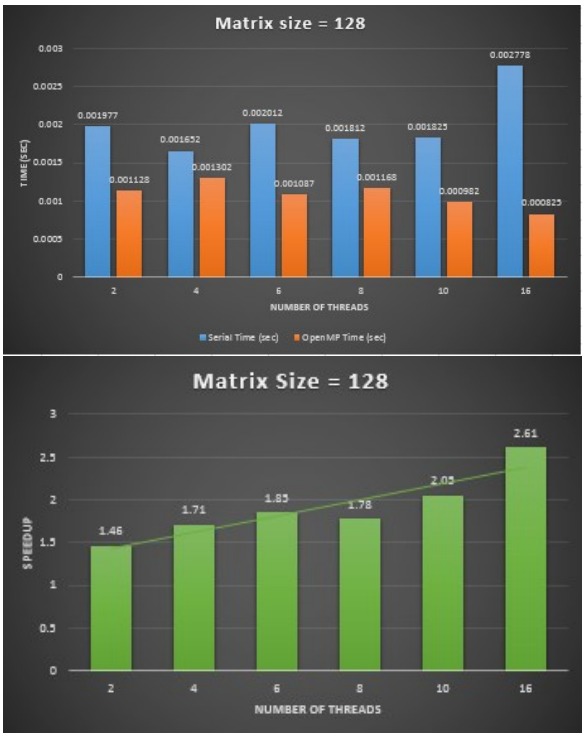
ج) خروجی حاصل از برنامه سریال

د) خروجی حاصل از برنامه موازی

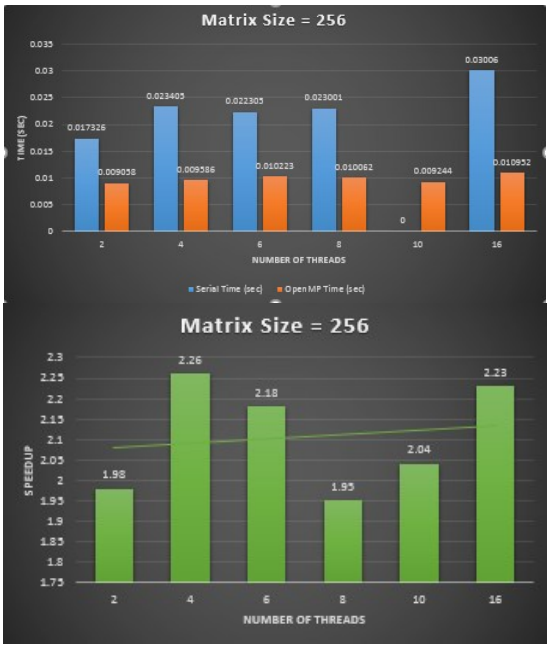
شکل(۳): نتایج حاصله از شکل سوم

۴- بررسی Efficiency در ضرب ماتریس‌ها با تغییر سایز ماتریس ها و تعداد Thread های موجود در موازی سازی

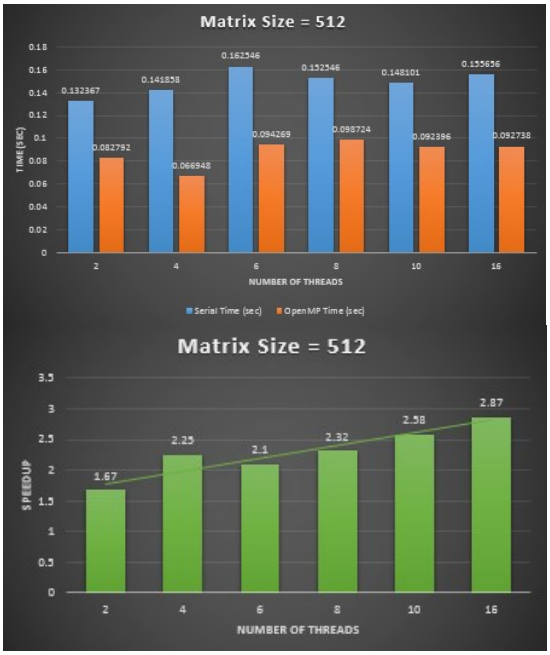
در این تمرین برنامه کامپیوتری ضرب ماتریسی دو ماتریس در دو حالت سریال و موازی (OpenMP) پیاده سازی شده است. با تغییر سایز ماتریس به ۱۲۸، ۲۵۶، ۵۱۲ و ۱۰۲۴ و همچنین با تغییر تعداد ریسه ها به ۲، ۴، ۶، ۸ و ۱۰ و ۱۶ به ازای هر سایز ماتریس، زمان اجرای هر برنامه را مجددا محاسبه می کنیم. بدین ترتیب میزان تاثیر تعداد ریسه ها در هر محاسبه مشخص می شود. جداول مربوط به این زمان سنجی ها با استفاده از نرم افزار Excel ایجاد شده و در ادامه آورده نشان داده شده است.



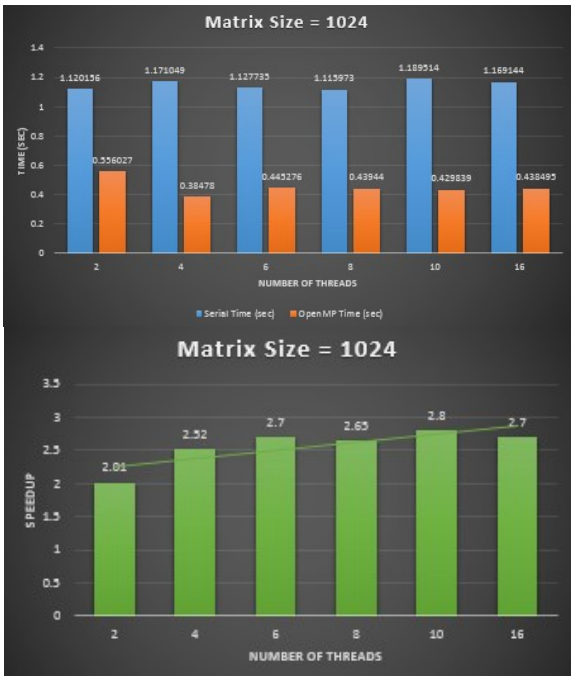
الف) مربوط به ابعاد ماتریس ۱۲۸



ب) مربوط به ابعاد ماتریس ۲۵۶



ج) مربوط به ابعاد ماتریس ۵۱۲



د) مربوط به ابعاد ماتریس ۱۰۲۴

در ادامه، میزان بهره وری یا efficieny نشان داده شده است. میزان بهره وری برابر با حاصل تقسیم SpeedUpبر تعداد ریسه ها است که نشان می دهد اولا Scalibility سیستم مورد نظر چقدر است و ثانیا اینکه چه تعداد ریسه به ازای هر سایز مناسب است با اینکه ریسه ها تا چه میزان مفید به کار گرفته شده اند. هدف بهینه سازی عملکرد سیستم است.

کدها و نتایج کامل این بخش در فایل "PartC" قرار دارد.

جدول۳: بررسی Efficiancy

Matrix Size	Number of Threads					
	2	4	6	8	10	12
128	0.9	0.505	0.495	0.3575	0.258	0.2025
256	0.99	0.6675	0.56	0.51375	0.294	0.19875
512	1.01	0.7725	0.57666	0.45125	0.381	0.258125
1024	1.005	0.88	0.616666	0.54125	0.429	0.25625

۵- بررسی تاثیر موازی سازی دو برنامه مختلف توسط دستورات OpenMP

برنامه داده شده را ابتدا به صورت سریال اجرا چند بار اجرا کردیم و زمان اجرای متوسط آن ۹,۷۲ ثانیه است.

سپس برنامه را به صورت موازی و با scheduleهای متفاوتی که در صورت سوال خواسته شده است اجرا کرده و نتایج را در جدول ۴ مشاهده می کنید. هر اجرا ۶ بار انجام شده که در جدول ۴ مقدار میانگین آن‌ها آورده شده است.

جدول ۴: زمان اجراهای کد مرحله ۴

نوع اجرا	میانگین زمان اجرا (s)	Gain
سریال	۹,۷۳۲	۱
Static	۵,۰۲۵	۱,۹۳
Static,1000	۳,۸۰۵	۲,۵۵
Static,2000	۳,۷۶۵	۲,۵۸
Dynamic,1000	۳,۵۸۰	۲,۷۲
Dynamic,2000	۳,۶۳۰	۲,۶۸

همان گونه که مشاهده می‌شود بهترین Gain بدست آمده مربوط به زمانی است که schedule را به صورت Dynamic استفاده کرده ایم که در حدود ۲,۷۲ برابر نتیجه سریع تر می‌دهد. در ادامه برنامه را کمی تغییر داده و سعی کرده ایم که زمان اجرای هر thread را به دست آوردیم. نتایج حاصله از scheduleهای گوناگون را در جدول ۵ مشاهده نمایید. هر اجرا ۶ بار انجام شده که در جدول ۵ مقدار میانگین آن‌ها آورده شده است.

جدول ۵: زمان تقریبی هر thread

نوع اجرا	زمان tread (s) 1	زمان tread (s)2	زمان tread (s) 3	زمان tread (s) 4
Static	۰,۸۵۰	۲,۳۷۵	۳,۷۲۵	۴,۹۴۵
Static 1000	۳,۳۴۵	۳,۴۰۲	۳,۴۳۳	۳,۴۷۱
Static 2000	۳,۱۹۱	۳,۳۴۶	۳,۴۵۸	۳,۵۳۷
Dynamic 1000	۳,۳۲۸	۳,۴۰۵	۳,۴۲۵	۳,۴۷۲
Dynamic 2000	۳,۲۲۸	۳,۳۲۹	۳,۴۳۵	۳,۵۳۳

همان گونه که در جدول ۵ مشاهده می کنید زمان اجرای هر thread در حالت Static بسیار نامتوازن می‌باشد که نشان دهنده عدم توازن بار در threadها می‌باشد. زمانی که برای Static پارامتر تعیین کنیم مشاهده می‌شود که به میزان قابل توجهی توازن بار برقرار می‌شود. زمانی که از Dynamic استفاده شود توازن بهتر می‌شود. یکی از دلایل کمتر شدن زمان اجرای کل برنامه، متوازن پخش شدن بار بین threadها می‌باشد.

کدها و نتایج دقیق‌تر را در فایل ”PartD“ مشاهده نمایید.

مراجع

[۱] مطالب ارایه شده در درس محاسبات با کارایی بالا دکتر صفری