گزارش پروژه
برنامه‌نویسی چندهسته‌ای

دکتر ممتازپور

حسین زارع‌دار ۹۶۳۱۰۳۵
نیمسال دوم ۹۹-۹۸

بستر اجرای پروژه

CPU: Intel Corei7 7700HQ (4 physical cores, 8 logical cores)
GPU: NVIDIA GeForce GTX 1050 (Notebook) (5 SMs, 4 GB of Memory)
Main Memory: 16GB
OS: Ubuntu 18.04

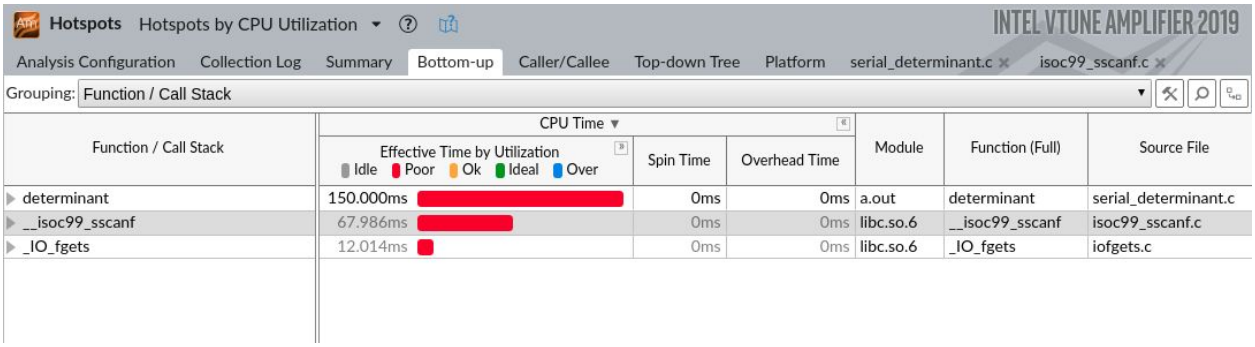
ورودی و خروجی برنامه

ورودی برنامه تعداد فایل است که در پوشه‌ی data_in در کنار فایل اجرایی قرار می‌گیرد. در هر فایل ۱ یا بیشتر از ۱ ماتریس به صورت row major ذخیره شده است که ماتریس‌ها با '\n' از یکدیگر جدا شده است.
خروجی برنامه، به ازای هر فایل ورودی، یک فایل در پوشه‌ی data_out ایجاد می‌شود که حاصل دترمینان ماتریس‌های فایل ورودی، در آن نوشته شده است.

نمونه‌ی کوچکی از ورودی و خروجی در کنار پروژه قرار دارد. ورودی دو فایل است که در هر فایل، ۳ ماتریس ۴ در ۴ قرار گرفته است.

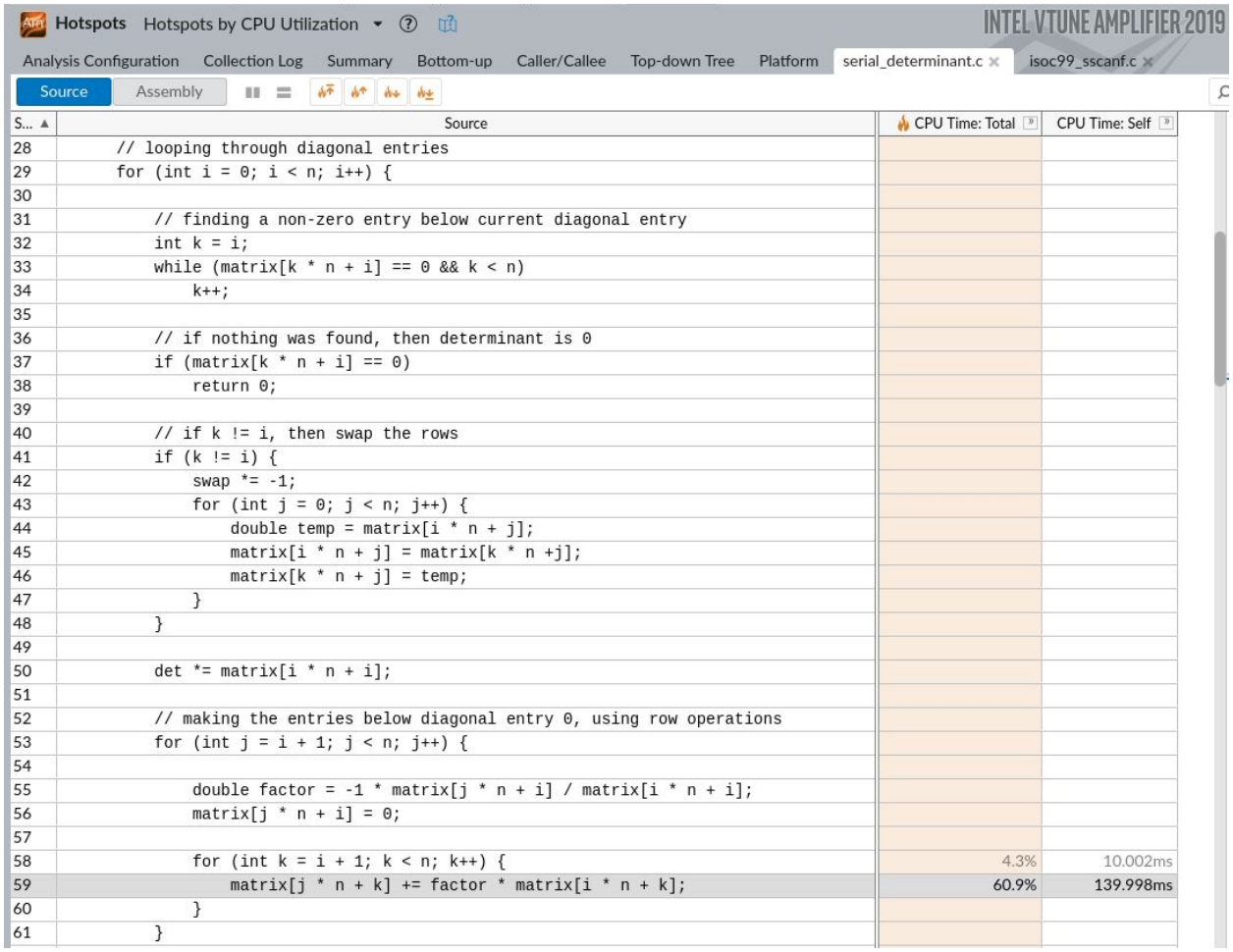
قدم اول: کد سریال

در قدم اول، کد سریال محاسبه‌ی دترمینان پیاده‌سازی شد.
نتیجه پروفایلینگ Hotspots به ازای ورودی ۲۰ فایل، هر کدام دارای ۱ ماتریس ۵۱۲ در ۵۱۲، به صورت زیر می‌باشد:



همانطور که مشاهده می‌شود، تقریبا یک‌سوم زمان اجرا به عملیات خواندن از فایل و ذخیره در حافظه، و بقیه زمان به محاسبات اختصاص دارد.

در قسمت محاسبات داریم:

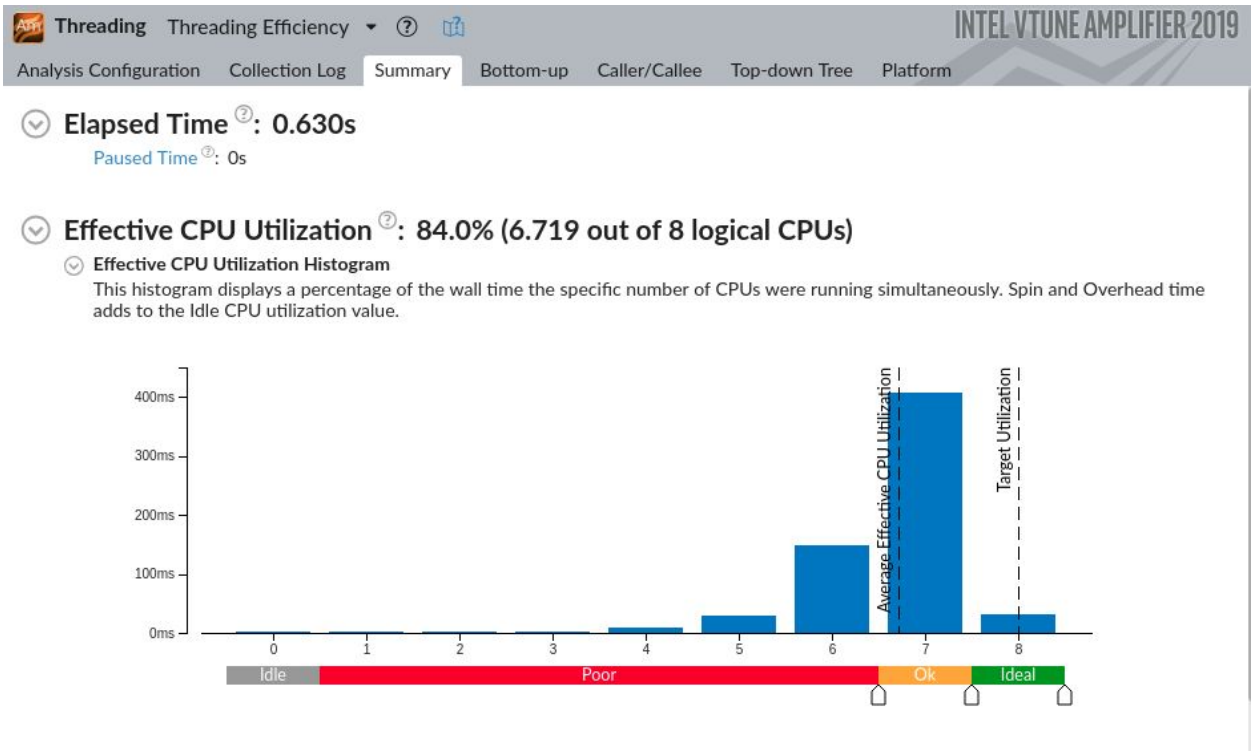


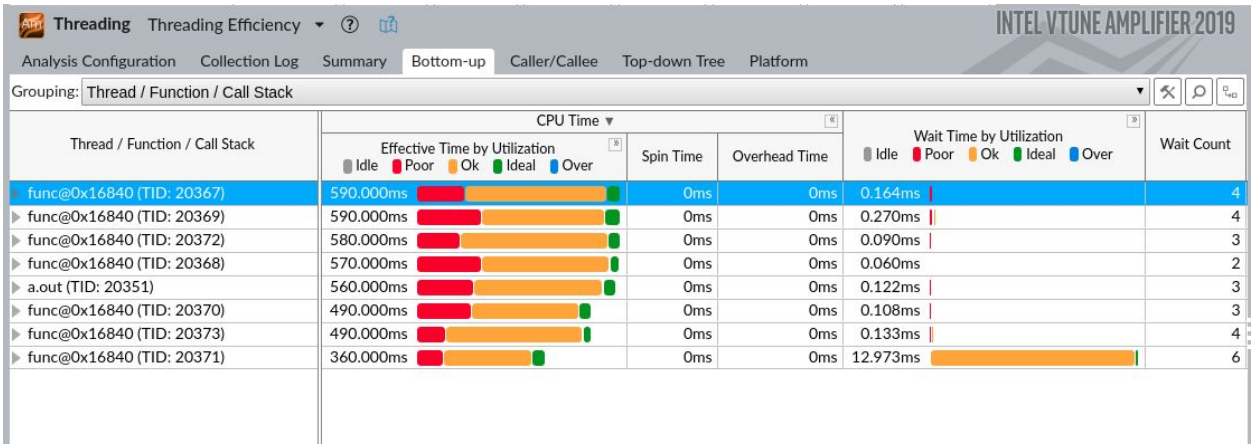
در محاسبه‌ی دترمینان به کمک تجزیه‌ی LU، حلقه‌ای که وظیفه‌ی انجام عملیات سطری را بر عهده دارد، بیشترین زمان اجرا را به خود اختصاص داده است.

قدم دوم: موازی‌سازی پردازش فایل‌ها به کمک OpenMP

در این قدم به کمک OpenMP، یک نخ به عنوان master، فایل‌های درون پوشه‌ی data_in را شناسایی می‌کند، سپس به ازای هر فایل یک task ایجاد کرده که نخ‌ها آن‌ها را انجام می‌دهند. هر نخ تسکی را برمی‌دارد و وظیفه‌ی خواندن یک فایل، محاسبه‌ی دترمینان ماتریس‌های درون آن فایل و نوشتن نتایجش را بر عهده می‌گیرد.

تحلیل Threading ازای ورودی ۲۰ فایل، هر کدام دارای ۱ ماتریس ۵۱۲ در ۵۱۲، به صورت زیر می‌باشد:





که مشاهده می‌شود، از هسته‌های پردازنده به شکل مناسبی استفاده شده است.

قدم سوم: محاسبه دترمینان به کمک پردازنده گرافیکی

در این قدم علاوه بر اقدامات مرحله قبل، یعنی از استفاده از OpenMP Task برای سپردن پردازش هر فایل به یک نخ پردازنده، هر نخ برای محاسبه‌ی دترمینان ماتریس‌های مربوط به فایل خود، از پردازنده گرافیکی بهره می برد.
برای محاسبه‌ی دترمینان در GPU، از همان تجزیه‌ی LU استفاده شده است، و برای مراحل مختلف آن، از توابع کتابخانه‌ی cuBLAS استفاده شده است.

مقایسه و تحلیل زمان اجرا

زمان اجرا ۳ نسخه‌ی برنامه (در حالت Release) از ابتدا تا انتهای آن (یعنی خواندن تمامی فایل‌ها، انجام محاسبات و نوشتن نتایج در فایل)، به ازای ورودی‌های مختلف به صورت زیر می‌باشد:

ورودی	Serial time	OMP Task		OMP Task + CUDA	
		time	speedup	time	speedup
۲۰۰ فایل، در هر فایل یک ماتریس ۲۵۶ در ۲۵۶	4.84s	1.43s	3.13	3.22s	1.50
۷۰ فایل، در هر فایل یک ماتریس ۵۱۲ در ۵۱۲	6.95s	2.60s	2.67	3.81	1.82
۵۰ فایل، در هر فایل یک ماتریس ۱۰۲۴ در ۱۰۲۴	23.14s	12.64s	1.83	10.23s	2.26
۲۰ فایل، در هر فایل یک ماتریس ۲۰۴۸ در ۲۰۴۸	59.6s	43.12s	1.28	21.45	2.77

همانطور که مشاهده می‌شود، در اندازه‌های کوچک ماتریس، استفاده از موازی‌سازی در CPU به کمک OMP Task، تسریع مناسبی را ایجاد می‌کند اما استفاده از GPU برای محاسبه‌ی دترمینان ماتریس‌ها، به علت سربار جابه‌جایی داده، تسریع را کاهش می‌دهد. ما با افزایش سایز ماتریس‌ها، مشاهده می‌کنیم که استفاده از GPU و CPU به طور همزمان، تسریع را به شکل خوبی افزایش می‌دهد.