در این گزارش به صورت کلی کار های بیان شده در دستور کار انجام شده اند و توضیحاتی در مورد آنها نوشته شده است و عکس های نیز از آنها گذاشته شده است.

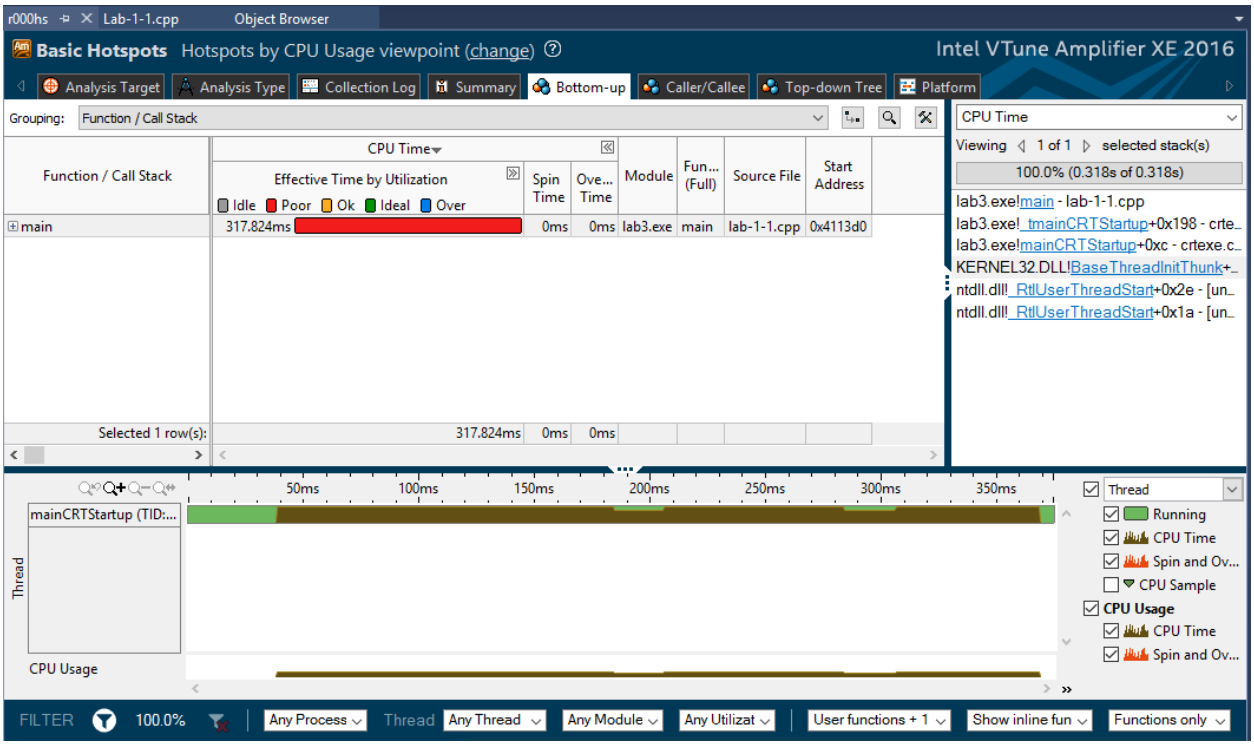
گام ۱)

ابتدا با استفاده از amplifier، hotspot های کد پیدا شده اند. پیش از انجام این کار تعداد تکرار حلقه بیرونی برابر با ۱ قرار داده شده چون این کار تاثیری در یافتن hotspot ها ندارد. همچنین مقدار VERYBIG نیز برابر با ۱۰۰۰۰ قرار داده شده است.

نتایج اولیه و پیش از موازی سازی برای حالت بیان شده در قسمت بالا به صورت زیر می باشد.

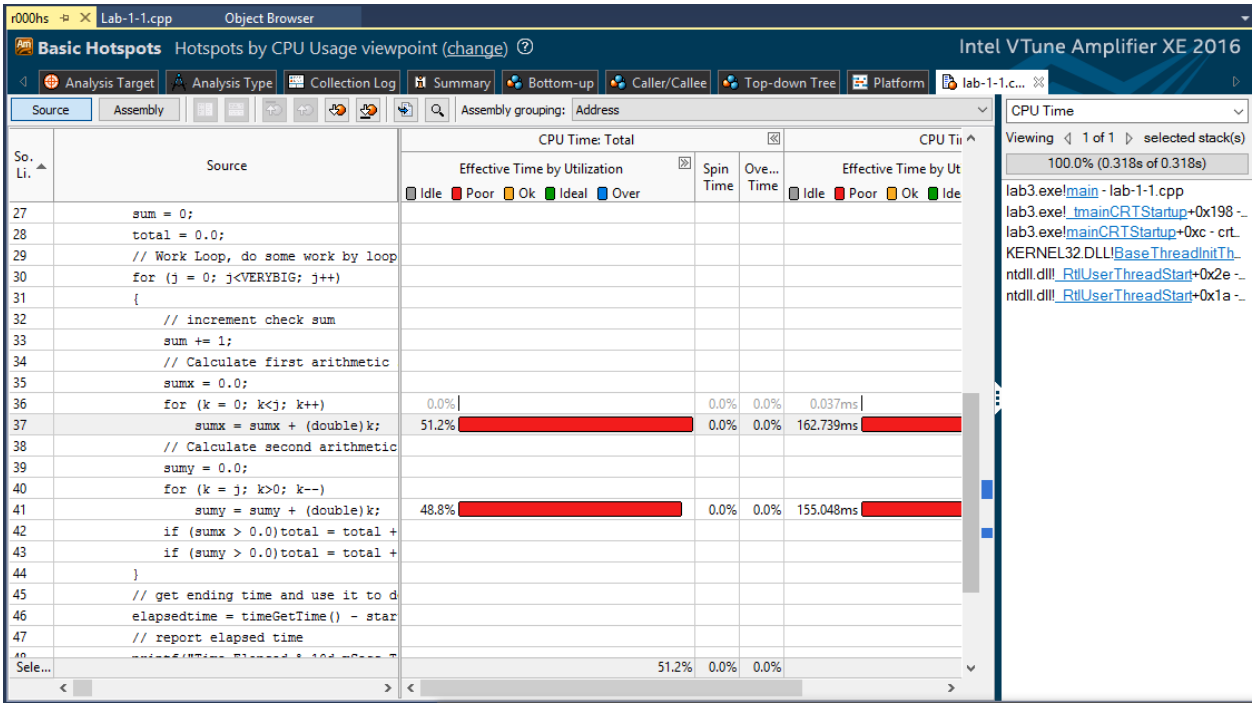
```
None Parallel Timings for 10000 iterations

Time Elapsed        140 mSecs Total = 26.104456 Check Sum = 10000
Press any key to continue
```

نتیجه خروجی amplifier برای یافتن hotspot ها به صورت زیر می باشد.



همان طور که دیده می شود زمان اجرا کد ۳۱۷ میلی ثانیه است.

Hotspot های پیدا شده در کد به صورت زیر می باشند.



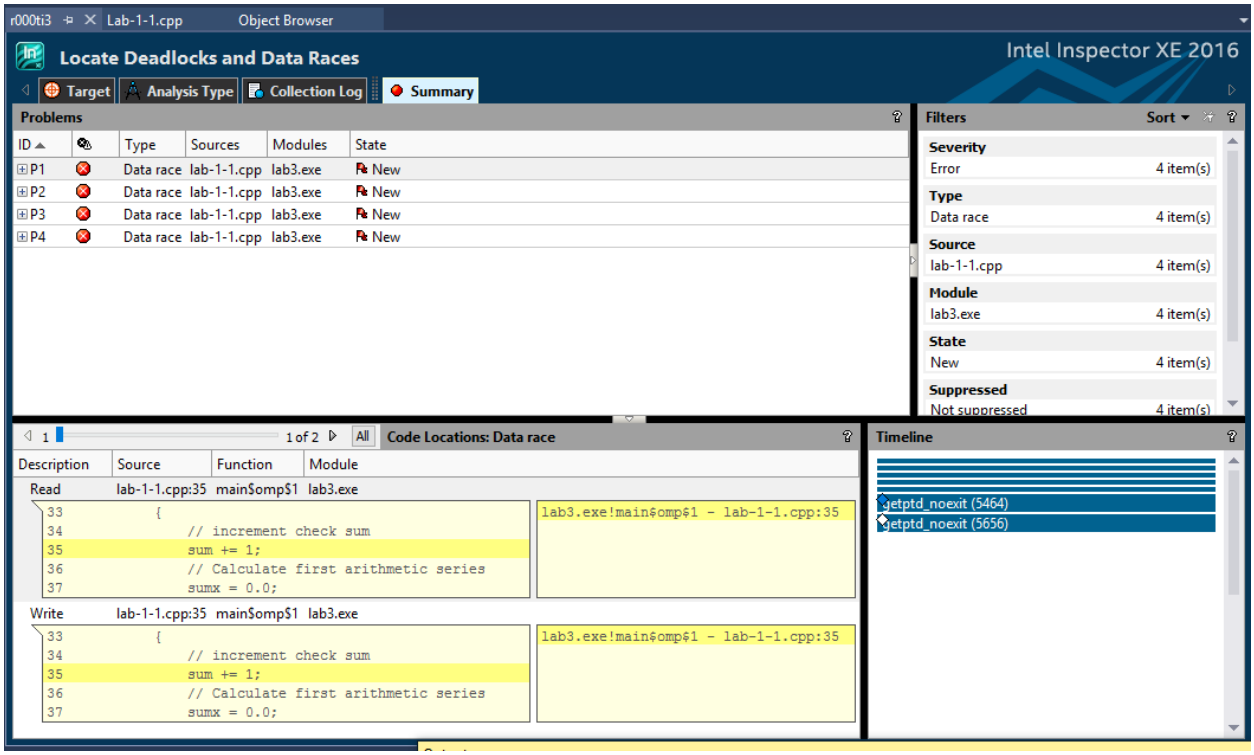
همان طور که مشخص است بیشترین زمان اجرا مربوط به قسمت هایی است که حلقه ریاضیاتی محاسبه می شود.

گام ۲)

خط #pragma omp parallel for به کد اضافه شده است تا موازی سازی را شروع کنیم.

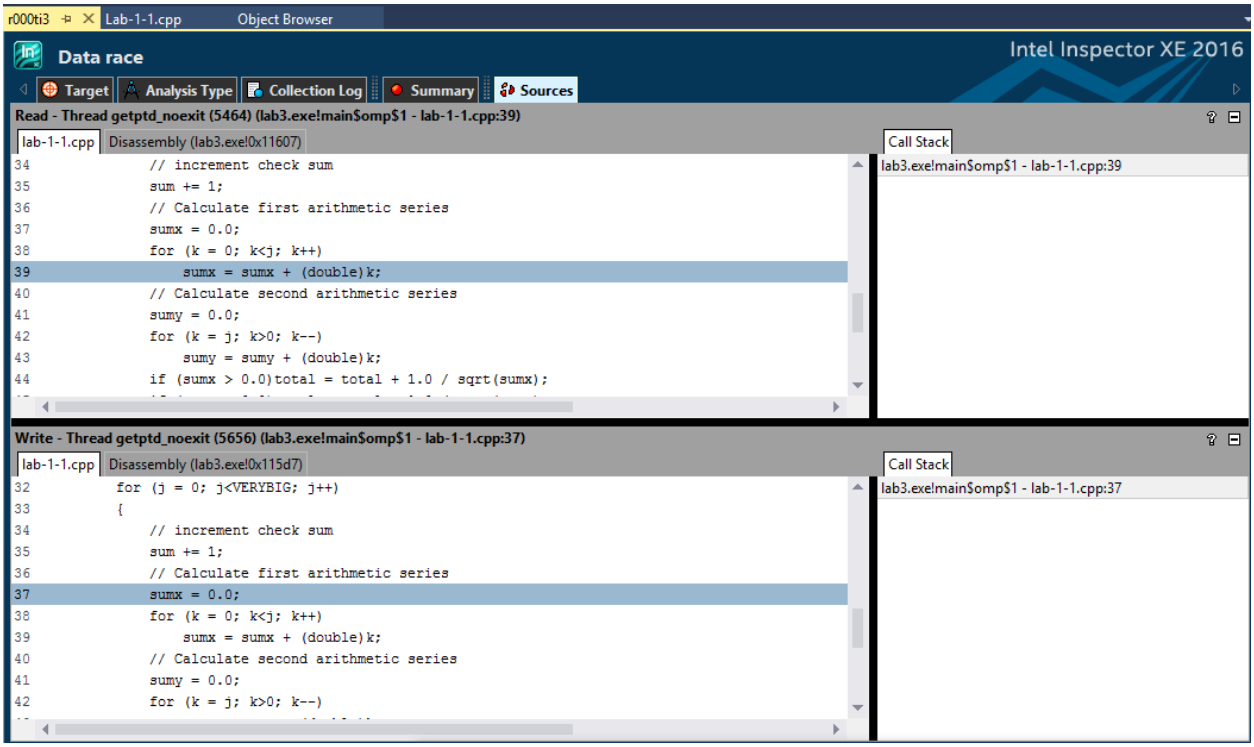
گام ۳)

همان طور که مشخص است این کد مشکل رقابت دارد. حالت اجرا در حالت debug قرار داده شده است. همچنین مقدار VERYBIG نیز به ۱۰۰۰ کاهش یافته است و حلقه بیرونی نیز صرفا یکبار اجرا می شود. سپس برنامه دوباره کامپایل شده است و شروع به پردازش آن با استفاده از ابزار inspector کرده ایم. خروجی آن به صورت زیر می باشد.



همان طور که مشخص است ۴ مشکل رقابت پیدا شده است که محل هر یک از آنها در کد نیز در پنجره پایین قابل مشاهده است.

پس از کلیک کردن روی p2 صفحه زیر را داریم.



که در آن تکه کد بالایی جایی را نشان می دهد که هنگام نوشتن رقابت رخ داده و تکه کد پایینی جایی را نشان می دهد که هنگام خواندن رقابت رخ داده است. از این دو کد مشخص است که متغیر sumx مشکلات رقابت دارد.

لیست کامل متغیر هایی که مشکل رقابت برایشان رخ می دهد به صورت زیر می باشد.

sum, total, sumx, sumy, k

دلیل این اتفاق نیز مشخصا این است که تمامی نخ ها از این متغیر ها به صورت همزمان استفاده می کنند. به این متغیر ها، متغیر های مشترک می گویند.

برای حل مشکل sumx , sumy, k خط زیر را به کد اضافه می کنیم.

#pragma omp parallel for private(sumx, sumy, k)

همچنین برای حل مشکل متغیر های sum و total که نمی توان آن ها را به صورت جداگانه در نظر گرفت و مقدارشان تجمیعی بدست می آید باید از دستور reduction استفاده کنیم که کد آن به صورت زیر می باشد.

#pragma omp parallel for private(sumx, sumy, k) reduction(+: sum, total)

با استفاده از این دستور هر نخ sum و total اخصاصی خواهد داشت و در انتها مقادیر این متغیر های اختصاصی جمع شده و در متغیر های نهایی با همین نام قرار می گیرد.

گام ۴)

آنالیز جدیدی را با استفاده از قسمت concurrency مربوط به amplifier انجام می دهیم.

با توجه به اینکه openmp گردش های حلقه را به صورت مرتب و به ترتیب شماره پخش می کند نا برابری بار کاری بین هسته ها خواهیم داشت چون کار حلقه های ابتدایی سبک تر می باشد. برای حل این مشکل کد زیر را اضافه می کنیم.

```
#pragma omp parallel for private( sumx, sumy, k ) reduction( +: sum, total ) schedule( dynamic, 2000 )
```

این کد باعث می شود که هر نخ در هر بار ۲۰۰۰ اجرا بگیرد سپس زمانی که کارش تمام شد برای می تواند بیاید و ۲۰۰۰ تای دیگر بگیرد و در این حالت تقسیم بار بسیار عادلانه تر خواهد بود.