۱ آنالیز کد سریال

ابتدا Hotspotهای برنامه را بررسی می کنیم تا متوجه شویم چه قسمتهایی از برنامه بیشترین زمان اجرا را به خود اختصاص دادهاند.

برای این کار با تبدیل مقدار VERYBIG به ۱۰ هزار و تعداد تکرارهای برنامه به ۱۰ آن را به کمک VTune می کنیم. پس از آنالیز شدن برنامه، با دو بار کلیک بر روی تابع main به سورس کد برنامه (شکل ۱) می رویم و خطوطی که بیشترین زمان اجرا را به خود اختصاص دادهاند را می بینیم.

for (j = 0; j <verybig; (double)="" (k="j," +="" arithmetic="" calculate="" check="" first="" for="" increment="" j++)="" k="" k++)="" k;="" k<j;="" second="" series="" sum="" sumx="sumx" sumy="0.0;" {="">0; k) sumy = sumy + (double) k; sumy = sumy + (double) k; for (k = j, k>0; k) sumy = sumy + (double) k; for (k = j, k>0; k) sumy = sumy + (double) k; for (sumx > 0.0) total = total + 1.0 / sqrt(sumx); if (sumy > 0.0) total = total + 1.0 / sqrt(sumy); // get ending time and use it to determine elapsed time</verybig;>			Lab-1-1.cpp ×	up Caller/Callee Top-down Tree Platform	ilysis Configu
for (j = 0; j <verybig; (double)="" (k="j," +="" arithmetic="" calculate="" check="" first="" for="" increment="" j++)="" k="" k++)="" k;="" k<j;="" second="" series="" sum="" sumx="sumx" sumy="0.0;" {="">0; k) for (k = j, k>0; k) if (sumx > 0.0) total = total + 1.0 / sqrt(sumx); if (sumy > 0.0) total = total + 1.0 / sqrt(sumy); // get ending time and use it to determine elapsed time</verybig;>					ource
Size	Self 🖹	CPU Time: Self			
// increment check sum sum += 1; // Calculate first arithmetic series sumx = 0.0; for (k = 0; k <j; (double)="" (k="j;" +="" arithmetic="" calculate="" for="" k="" k++)="" k;="" second="" series="" sumx="sumx" sumy="0.0;">0; k) sumy = sumy + (double) k; sumy = sumy + (double) k; for (k = j; k>0; k) sumy = sumy + (double) k; for (k = j; k>0; k) sumy = sumy + (double) k; for (sumy > 0.0) total = total + 1.0 / sqrt(sumx); if (sumy > 0.0) total = total + 1.0 / sqrt(sumy); // get ending time and use it to determine elapsed time</j;>					
Sum += 1;					
// Calculate first arithmetic series sumx = 0.0; for (k = 0; k <j; (double)k;="" (k="j;" +="" arithmetic="" calculate="" for="" k="" k++)="" second="" series="" sumx="sumx" sumy="0.0;">0; k) sumy = sumy + (double)k; 2 if (sumx > 0.0) total = total + 1.0 / sqrt(sumx); if (sumy > 0.0) total = total + 1.0 / sqrt(sumy); // get ending time and use it to determine elapsed time</j;>					
5					
for (k = 0; k <j; 7<="" k++)="" td=""><td></td><td></td><td></td><td>series</td><td></td></j;>				series	
7					
8					
9	318ms	46.318	52.4%		
0 for (k = j; k>0; k) 1 sumy = sumy + (double)k; 47.6% 42. 2 if (sumx > 0.0)total = total + 1.0 / sqrt(sumx); 3 if (sumy > 0.0)total = total + 1.0 / sqrt(sumy); 4 } 5 // get ending time and use it to determine elapsed time				series	
1					
2					
3 if (sumy > 0.0)total = total + 1.0 / sqrt(sumy); 4 } 5 // get ending time and use it to determine elapsed time	068ms	42.068	47.6%		
4 } 5 // get ending time and use it to determine elapsed time				1.0 / sqrt(sumx);	
5 // get ending time and use it to determine elapsed time				1.0 / sqrt(sumy);	
77 9-1					
6 elapsedtime = timeGetTime() - starttime;				etermine elapsed time	
				ttime;	
7 // report elapsed time					

شكل ١: نتايج آناليز با VTune

۲ موازی سازی به کمک OpenMP

پس از مشاهده قسمتهایی از برنامه که زمان اجرای آنها طولانی تر است، سعی می کنیم با موازی سازی این بخشها تسریع بگیریم. با گذاشتن خط زیر پیش از حلقه work، اجراهای آن را موازی می کنیم.

#pragma omp parallel for

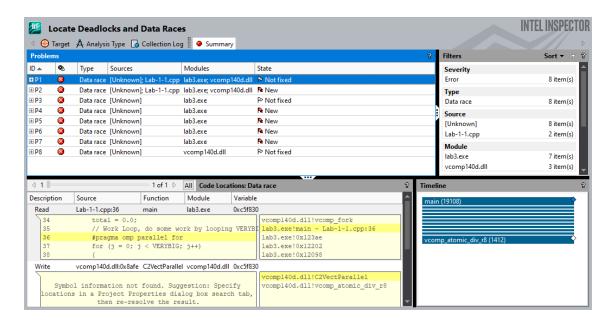
مشاهده می شود که در این حالت زمان اجرای برنامه حتی بیشتر شده است و برنامه به درستی اجرا نمی شود (مقدار متغیرهای sum و total درست نیست).

۴

۳ دیباگ و رفع خطاها

با ابزار Inspector و كاهش دادن مقدار VERYBIG به هزار، برنامه را تحليل مي كنيم.

همانطور که در شکل ۲ میبینیم، برنامه دارای شرایط مسابقه برای متغیرهای مشترک (sumx ،total ،sum، k) است.



شکل ۲: وجود شرایط مسابقه برای متغیرهای مشترک بین نخها

با استفاده از خط زیر، متغیرهایی که می توانند خصوصی باشند را خصوصی می کنیم. همچنین متغیرهایی که باید مقدار آنها بین همه نخها مشترک باشد را می توانیم با استفاده از عبارت reduction جلوی شرایط مسابقه آن را بگیریم.

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

۴ تنظیم و سرعت بخشیدن به برنامه OpenMP

با آنالیز برنامه به کمک VTune مطابق شکل ۳، مشاهده میکنیم که کار به صورت نامتوازن بین نخها پخش شده است. در واقع برخی نخها زودتر کارشان به اتمام میرسد و به ناچار منتظر دیگر نخها میمانند. همچنین در شکل ۴ میبینیم که مدت زمان بسیار کمی هر ۸ نخ همزمان با هم فعال هستند (ستون سمت

CPU Sample

☐ ↓↑ Transitions

✓ CPUUItilization

▼ User functions + 1 ▼ Functions only ▼ Show inline functic ▼

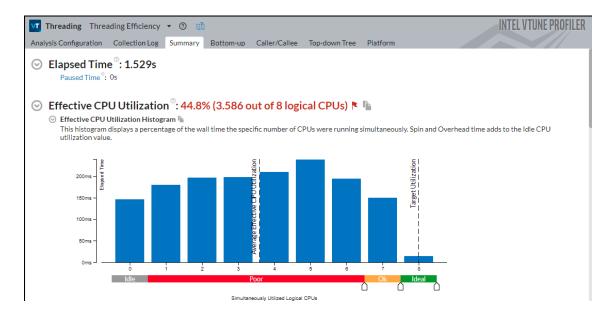
INTEL VTUNE PROFILER ▼ Threading Threading Efficiency ▼ ② 📆 Analysis Configuration Collection Log Summary Bottom-up Caller/Callee Top-down Tree CPU Time Grouping: Thread / Function / Call Stack Viewing 1 of 4 ► selected stack(s) CPU Time 73.3% (1.012s of 1.381s) Thread / Function / Call Stack Effective Time by Utilization

Idle Poor Ok Ideal Over Lab3.exe!func@0x140011a00 - [unknown source file] Imbalance or Serial Spinning Lock Cont VCOMP140D.DLL!func@0x18000186c+0x8 - [unknown so.. VCOMP140D.DLL!func@0x1800017c0+0x70 - [unknown s... ▶ Thread (TID: 1736) 1.012s VCOMP140D.DLL!func@0x1800097c0+0x59 - [unknown s... Thread (TID: 12564) KERNEL32.DLL!BaseThreadInitThunk+0x13 - [unknown so... ▶ Thread (TID: 14220) 1.142s 0s ntdll.dll!RtlUserThreadStart+0x20 - [unknown source file] ► Thread (TID: 20256) 1.326s 0s Thread (TID: 20980) 0.594s 0s ▶ Thread (TID: 21152) 0.369s Os ▶ Thread (TID: 1856) 0.192s 0s ρ: **+** Thread ~ Running ✓ ☐ Waits Thread (TID: 21152) CPU Time Thread (TID: 1856) Spin and Overhead...

راست) و این به معنی استفاده نامناسب از تمام توان پردازشی پردازنده است.

شکل ۳: پخش نامتوازن کار بین نخها

Any Process V Thread Any Thread V Any Module



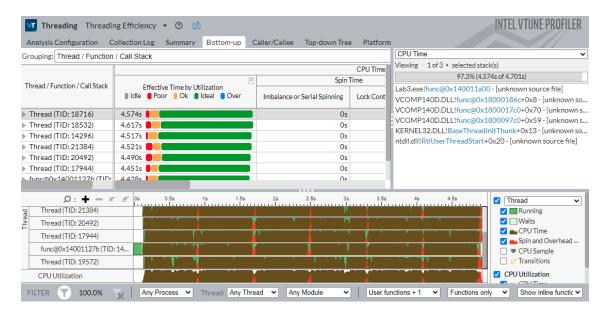
شکل ۴: عدم استفاده از تمام توان پردازشی پردازنده به علت پخش نامتوازن کارها

این پخش نامتوازن به علت محاسبات بیشتر نخهای پایانی در حلقه k است. با استفاده از عبارت زیر میتوانیم پخش کارها را متوازن کنیم.

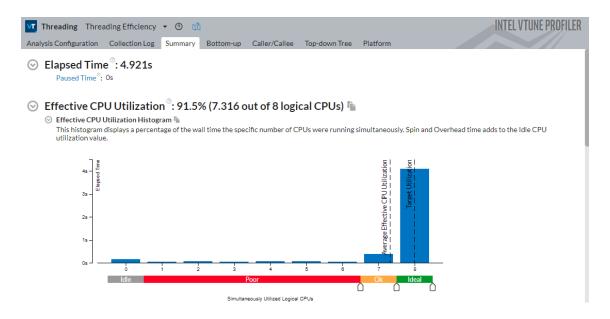
#pragma omp parallel for private(sumx, sumy, k) \
reduction(+:sum, total) schedule(dynamic, 2000)

func@0x140011276 (TID: 11...

با آنالیز مجدد کار کرد نخها با استفاده از VTune، میبینیم که پخش کارها بسیار متوازن تر است و به بهرهوری ایده آل بسیار نزدیک تر شده ایم (شکلهای δ و δ)



شکل ۵: پخش بسیار بهتر کارها پس از استفاده از schedule(static) و با chunk size برابر ۲۰۰۰



شکل ۶: نزدیک شدن به میزان بهرهوری ایدهآل پردازنده