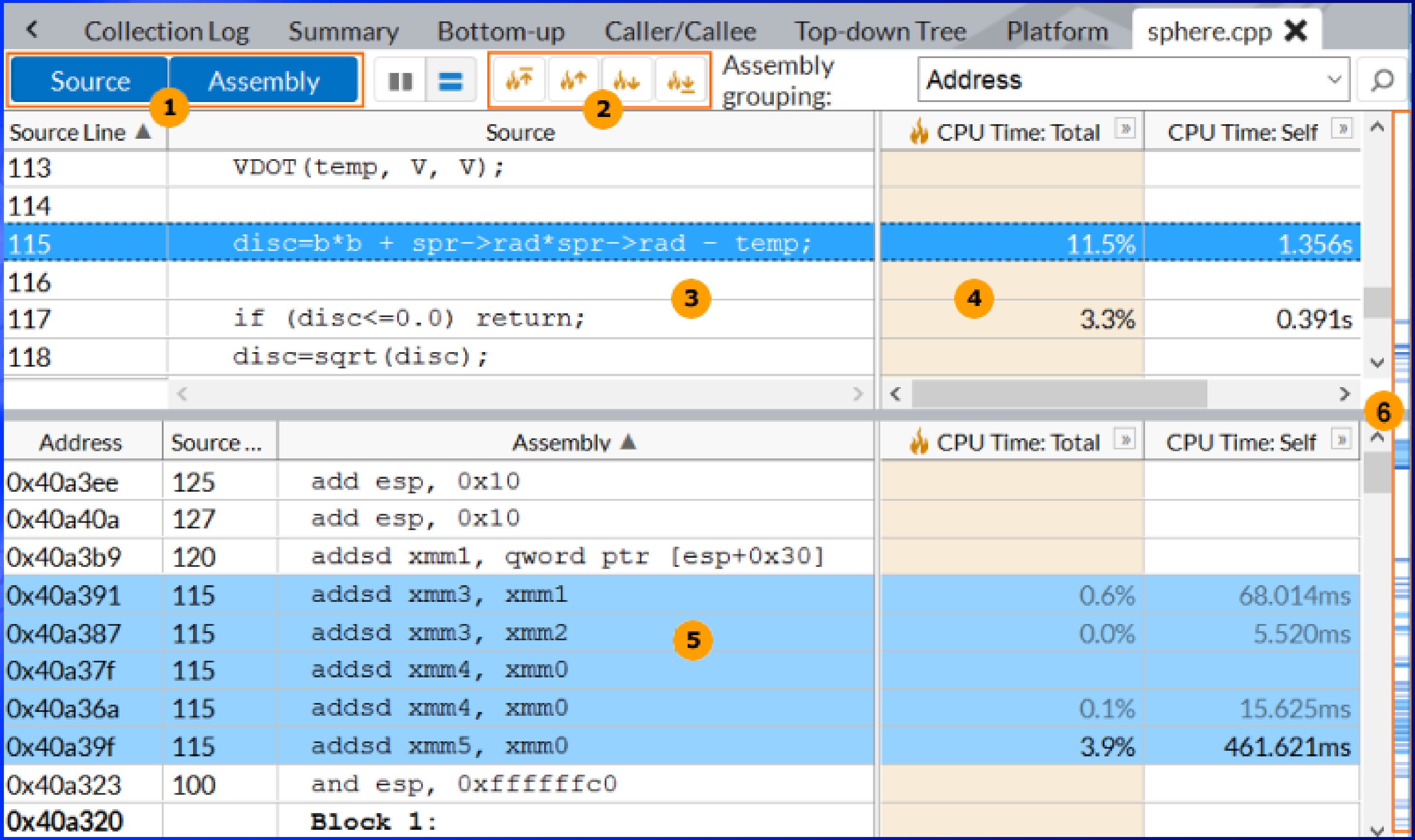
Workload Analysis

(Stage 2)

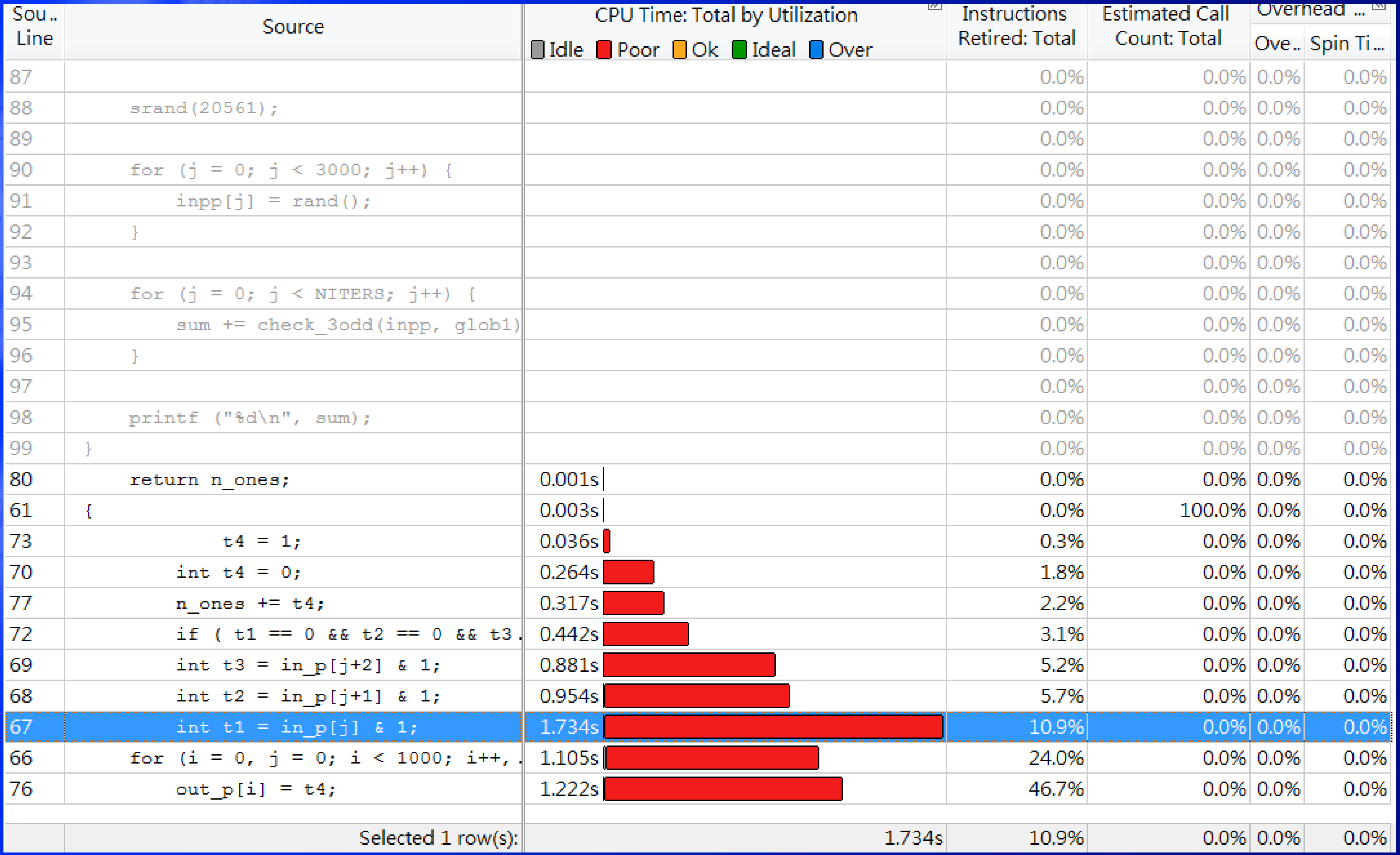
- 請嘗試利用 VTune profiler 或其他工具來分析應用程式的hotspot



Source Line ▲	Source	CPU Time: Total	CPU Time: Self
113	`VDOT(temp, V, V);`		
114			
115	`disc=b*b + spr->rad*spr->rad - temp;`	11.5%	1.356s
116			
117	`if (disc<=0.0) return;`	3.3%	0.391s
118	`disc=sqrt(disc);`		

Address	Source ...	Assembly ▲	CPU Time: Total	CPU Time: Self
0x40a3ee	125	`add esp, 0x10`		
0x40a40a	127	`add esp, 0x10`		
0x40a3b9	120	`addsd xmm1, qword ptr [esp+0x30]`		
0x40a391	115	`addsd xmm3, xmm1`	0.6%	68.014ms
0x40a387	115	`addsd xmm3, xmm2`	0.0%	5.520ms
0x40a37f	115	`addsd xmm4, xmm0`		
0x40a36a	115	`addsd xmm4, xmm0`	0.1%	15.625ms
0x40a39f	115	`addsd xmm5, xmm0`	3.9%	461.621ms
0x40a323	100	`and esp, 0xffffffc0`		
0x40a320		`Block 1:`		

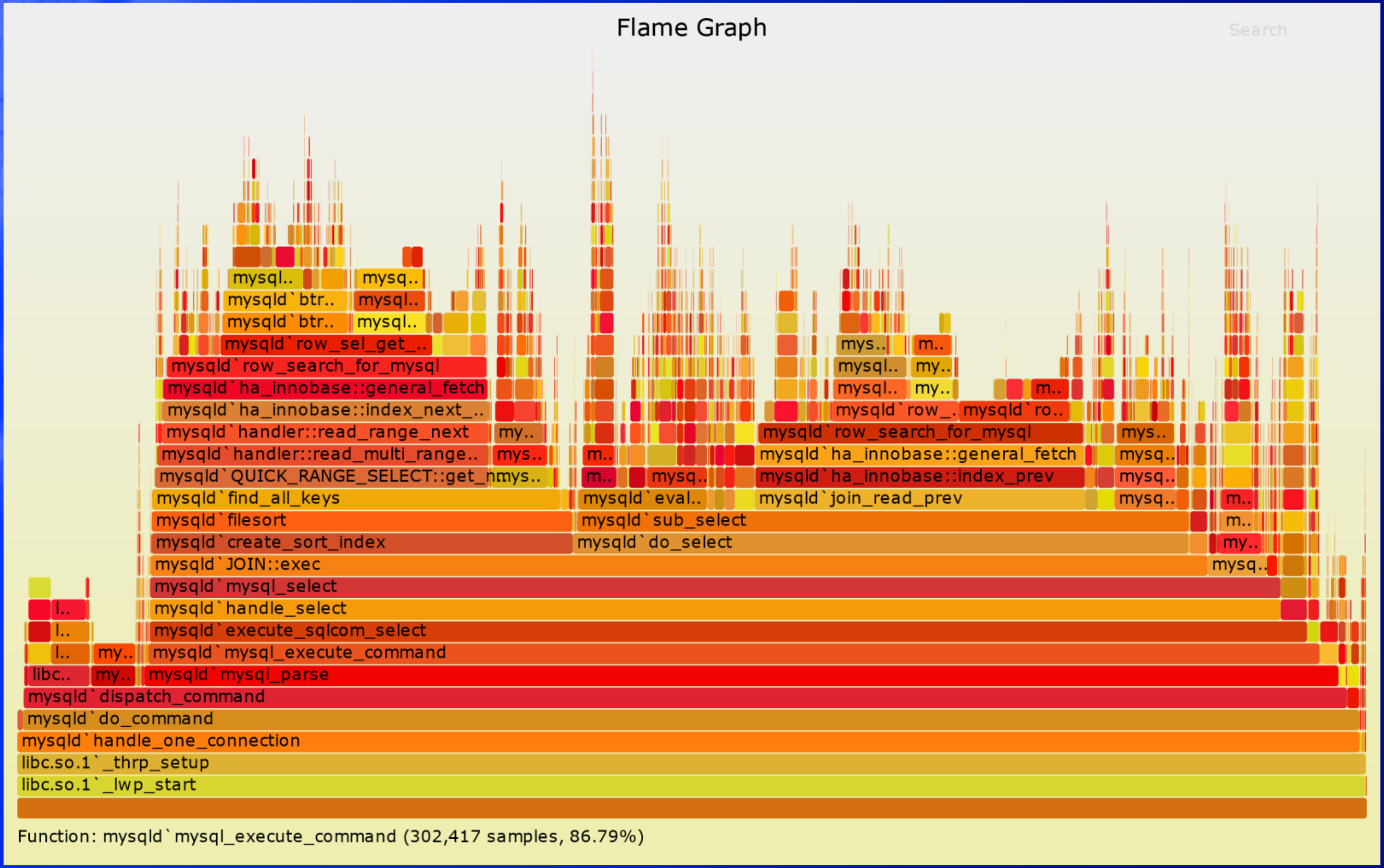
利用VTune可以檢視source code裡每個statement的執行時間、event發生次數等訊息，瞭解是程式裡哪個部分執行較慢，分析其原因，並進一步提出加速的策略。



Sou.. Line	Source	CPU Time: Total by Utilization ▨ Idle 🟥 Poor 🟧 Ok 🟩 Ideal 🟦 Over	Instructions Retired: Total	Estimated Call Count: Total	Overhead ... Ove..	Spin Ti...
87			0.0%	0.0%	0.0%	0.0%
88	`srand(20561);`		0.0%	0.0%	0.0%	0.0%
89			0.0%	0.0%	0.0%	0.0%
90	`for (j = 0; j < 3000; j++) {`		0.0%	0.0%	0.0%	0.0%
91	`inpp[j] = rand();`		0.0%	0.0%	0.0%	0.0%
92	`}`		0.0%	0.0%	0.0%	0.0%
93			0.0%	0.0%	0.0%	0.0%
94	`for (j = 0; j < NITERS; j++) {`		0.0%	0.0%	0.0%	0.0%
95	`sum += check_3odd(inpp, glob1)`		0.0%	0.0%	0.0%	0.0%
96	`}`		0.0%	0.0%	0.0%	0.0%
97			0.0%	0.0%	0.0%	0.0%
98	`printf ("%d\n", sum);`		0.0%	0.0%	0.0%	0.0%
99	`}`		0.0%	0.0%	0.0%	0.0%
80	`return n_ones;`	0.001s	0.0%	0.0%	0.0%	0.0%
61	`{`	0.003s	0.0%	100.0%	0.0%	0.0%
73	`t4 = 1;`	0.036s	0.3%	0.0%	0.0%	0.0%
70	`int t4 = 0;`	0.264s	1.8%	0.0%	0.0%	0.0%
77	`n_ones += t4;`	0.317s	2.2%	0.0%	0.0%	0.0%
72	`if ( t1 == 0 && t2 == 0 && t3 .`	0.442s	3.1%	0.0%	0.0%	0.0%
69	`int t3 = in_p[j+2] & 1;`	0.881s	5.2%	0.0%	0.0%	0.0%
68	`int t2 = in_p[j+1] & 1;`	0.954s	5.7%	0.0%	0.0%	0.0%
67	`int t1 = in_p[j] & 1;`	1.734s	10.9%	0.0%	0.0%	0.0%
66	`for (i = 0, j = 0; i < 1000; i++, .`	1.105s	24.0%	0.0%	0.0%	0.0%
76	`out_p[i] = t4;`	1.222s	46.7%	0.0%	0.0%	0.0%
	Selected 1 row(s):	1.734s	10.9%	0.0%	0.0%	0.0%

Flame Graph

- 嘗試使用Flame graph做觀察 (VTune也有支援Flame graph)
- https://www.brendangregg.com/flamegraphs.html



Function: mysqld`mysql_execute_command (302,417 samples, 86.79%)

- 利用sampling的方式量測
  - CPI
  - Cache miss相關的events
  - Branch相關的events
  - Any others ...

- 目標：針對主要的函式，找出其執行較久的原因。
- 是否有解決方法?

(檢視數據是否有特別之處？哪些code fragments是hotspot?)

- 如果VTune因故無法執行或量測，應找尋其他performance tool代替。(如: Linux perf、Valgrind、Oprofile)

- 請觀察是否有使用SIMD技術，加速程式執行效能的機會?
- 是否有使用compiler directive，加速程式執行效能的機會?
  - Intel compiler: https://www.intel.com/content/www/us/en/develop/documentation/fortran-compiler-oneapi-dev-guide-and-reference/top/language-reference/directive-enhanced-compilation/general-compiler-directives.html
- 改寫部分程式碼?
- 其他可能的優化想法?

Presentation

- Presentation (應該盡量讓數據或你的結論可以友善的呈現，使聽眾較容易地了解)
  - Describe your benchmark (簡易描述)
  - Evaluation environment
  - 找尋hotspot、優化機會的步驟與方法
  - 可能被優化的code fragments、是否有SIMD的機會、預計/嘗試優化的方法、…
  - Report your data and status
  - TO DO
- Deadline: Dec. 12, 2023 (線上同步、口頭報告)