ループで呼び出される大きな関数を分割して命令キャッシュを最適化する

この記事は、インテル® デベロッパー・ゾーンに公開されている「Split huge function if called by loop for best utilizing Instruction Cache」の日本語参考訳です。

命令キャッシュミスは、フロントエンドのストールを引き起こす重大な問題です。通常、アプリケーションに分岐予測ミスが多発する大きなホットコード領域が含まれる場合、ICache 予測ミスによるストールが頻発し、ホットコード領域が呼び出される回数に応じてこのストールの回数も増えます。この問題を解決するには、インテル® C/C++ コンパイラーのプロファイルに基づく最適化 (PGO) を利用して、プログラムのバイナリーベースの実行パスを生成します。

分岐と関係なく生じる ICache ミスに、関数が 4KB のページ境界をまたいだで配置される場合があります。数百行からなる大きな関数を記述し、この関数をループから頻繁に呼び出すプログラムを インテル® VTune™ Amplifier XE で解析してみたところ、ICache ミスが頻発していることが分かりました。キャッシュラインは 64 バイトで、命令のフェッチは 4KB ページ内を参照します (32KB の ICache では合計 8 ウェイ)。関数が 4KB よりも大きい場合はページ境界をまたぐため、キャッシュミスとなり、メモリーアクセスが発生します。

このため、ループでは大きな関数を呼び出さないようにします。ホット領域が頻繁に呼び出される場合は、この関数を小さい関数に分割し、キャッシュミスを減らします。loop-function の使用を loop-func1、loop-func2、…、loop-funcN にすると良いでしょう。

変更前:

```
ループ
  関数:
領域 1
  領域 2
  …
  領域 N
ループ終了
```

変更後:

```
ループ
  Func1 の呼び出し
ループ
  Func2 の呼び出し
…
ループ
  FuncN の呼び出し
```

テストに使用できる大きな関数 (数百行からなる 4KB を超える関数) が手元にないので、簡単なテストケースのペアを作成し (オリジナルの関数は 1 キャッシュラインのサイズである 64 バイトを超えます)、インテル® VTune™ Amplifier XE のレポートを利用してパフォーマンス・データを比較してみました。使用したテストコードは、記事の最後にあります。

```
# g++ -g test_loop_with_long_func.cpp –o test_loop_with_long_func
# g++ -g test_loops_with_small_funcs.cpp –o test_loops_with_small_funcs
```

上記のコマンドでソースファイルをコンパイルしたら、インテル® VTune™ Amplifier XE でテストします (ICACHE.MISSES イベントを使用します)。

ICACHE.MISSES イベントは、ICache ミスになりメモリーアクセス要求が発生したすべての命令フェッチをカウントします。

ケース 1: 1 つのループで大きな関数を呼び出す

```
# amplxe-cl -collect-with runsa -knob event-
config=CPU_CLK_UNHALTED.THREAD,INST_RETIRED.ANY,ICACHE.MISSES:sa=5000 --
 ./test_loop_with_long_func

#amplxe-cl -R summary -r r000runsa/
```

実行結果

経過時間: 10.324
CPU 時間: 10.119
CPI レート: 0.395

イベント結果

ハードウェア・イベント・タイプ	ハードウェア・イベント・カウント:Self	ハードウェア・イベント・サンプル・カウント: Self	サンプルあたりのイベント数
CPU_CLK_UNHALTED.THREAD	38292057438	19146	2000003
INST_RETIRED.ANY	96828145242	48414	2000003
CPU_CLK_UNHALTED.REF_TSC	34322051483	17161	2000003
ICACHE.MISSES	540000	108	5000

ソースビューにドリルダウンしてコード中の ICache ミスを確認します。

Source | Assembly | Assembly grouping: Address

S. Li.	Source	Hardware Event Count by Hardware Event Type: CPU_CLK...	INST_RETI...	CPU_CLK_...	ICACHE.MISSES
46	// memory copy_2				
47	for(i=0;i<NUM;i++) {	14,000,021	42,000,063	20,000,030	0
48	for(j=0;j<NUM;j++) {	96,000,144	174,000,261	92,000,138	5,000
49	a[i][j]=b[i][j]*PI;	320,000,480	676,001,014	282,000,423	0
50	}				
51	}				
52					
53	// multiply				
54	for(i=0;i<NUM;i++) {	16,000,024	20,000,030	18,000,027	5,000
55	for(j=0;j<NUM;j++) {	652,000,978	1,782,002,...	548,000,822	0
56	c[i][j] = 0.0;	84,000,126	196,000,294	68,000,102	0
57	for(k=0;k<NUM;k++) {	6,118,009,...	13,780,02...	5,582,008,...	10,000
58	c[i][j] += a[i][k]	29,192,043...	76,750,11...	26,068,03...	20,000

ケース 2: n 個のループで n 個のサブ関数を呼び出す

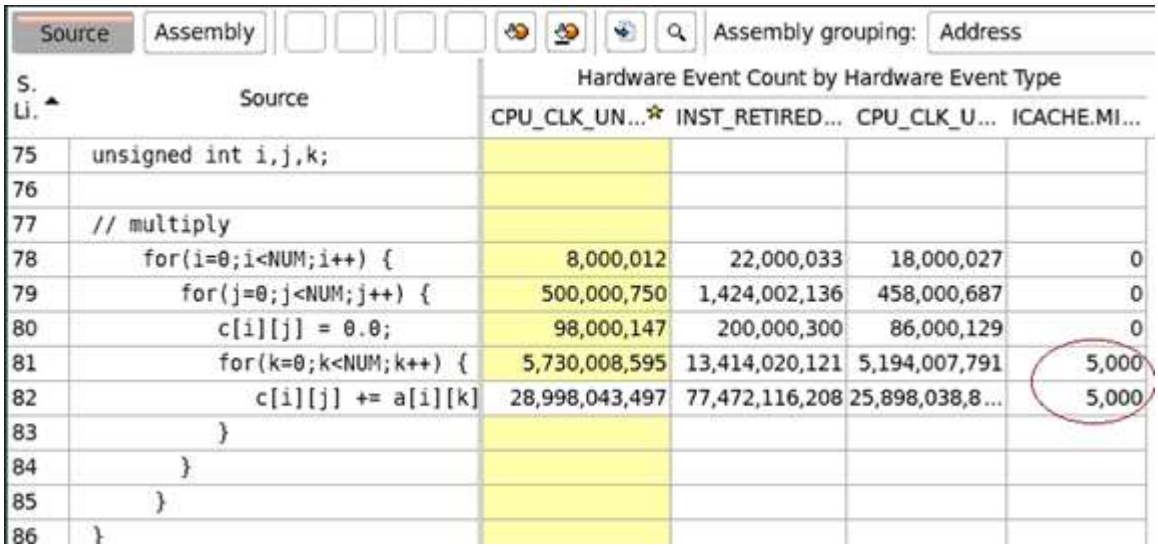
```
# amplxe-cl -collect-with runsa -knob event-
config=CPU_CLK_UNHALTED.THREAD,INST_RETIRED.ANY,ICACHE.MISSES:sa=5000 --
 ./test_loops_with_small_funcs

# amplxe-cl -R summary -r r001runsa/
```

実行結果

経過時間: 10.112
CPU 時間: 9.911
CPI レート: 0.387

イベント結果

ハードウェア・イベント・タイプ	ハードウェア・イベント・カウント:Self	ハードウェア・イベント・サンプル・カウント: Self	サンプルあたりのイベント数
CPU_CLK_UNHALTED.THREAD	37498056247	18749	2000003
INST_RETIRED.ANY	96824145236	48412	2000003
CPU_CLK_UNHALTED.REF_TSC	33618050427	16809	2000003
ICACHE.MISSES	265000	53	5000



テストに使用したサンプルコード:

test_loop_with_long_func.cpp (1.3 KB)
test_loops_with_small_funcs.cpp (1.81 KB)

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください