新しい最適化レポートを使用して、コンパイラーを最大限に活用する：

Kiyo Sugawara

September 2015

Rev 2.1

Optimization Notice

このセッションの目的

インテル® Parallel Studio XE 2016 に統合され、強化されたコンパイラーの最適化レポートについて学ぶ：

提供される情報を制御

パフォーマンスを向上するため、コンパイラーの最適化レポートの情報を理解し、チューニングに役立てる

isus の記事をご覧ください。

Optimization Notice

新しい最適化レポートとは？

利用者からのフィードバック ……

- ベクトル化や他のレポートで提供されるメッセージは理解するのが難しい
- 暗号のようなメッセージ
- ループの最適化メッセージが、複数のレポートに分割されている
- コンパイラーによって生成される複数バージョンのループは混乱を招く
- コードがインライン展開されると、ソースコードに対応付けるのが困難
- 1 つの大きなレポートが生成される；並列ビルドには不向き

16.0 コンパイラーにおけるレポートの目標

解り易さを改善

- 見やすく、理解しやすいレポートを生成する
  - 単一の統合されたレポート
  - ループベースのレポート
- プログラマーが対処しやすい情報を提供
- 必要な情報を選択し易いようにする
- 出力モードの範囲を拡張

利用しやすいレポートを生成

- テキスト・ファイルで生成
- アセンブリー・リスト中にも生成
- Microsoft* Visual Studio* IDE で利用できるようにする
- 他のインテル・ソフトウェア・ツールで利用できるようにする

概要

インテル® コンパイラー　バージョン 16.0 に適用

- C/C++ と Fortran
- Windows*、Linux* および OS X*

（スライドを読み易くするため、以降各 OS でスペルが類似するオプションは繰り返して明記していないことがあります。特に明記されない限り、オプションはすべての OS に適用されます）

主なオプション：

/Qopt-report[:N] (Windows)、-qopt-report[=N] (Linux と OS X)

N = 1-5　（詳細レベルを選択。デフォルトは N = 2）

/Qopt-report-phase:str[,str1,…]

str = loop, par, vec, openmp, ipo, pgo, cg, offload, tcollect, all

/Qopt-report-file: stdout | stderr | ファイル名

/Qdiag-message-catalog-

出力されるレポート

デフォルトでは、レポートは標準出力ではなくテキスト形式で **ファイル** に出力される

- ファイルの拡張子は、.optrpt です。ファイル名はオブジェクト・ファイルと同じ名称になる
- オブジェクト・ファイルごとに 1 つのレポートが、オブジェクトと同じディレクトリーに作成される
- 新規作成もしくは上書き（ファイルへの追加は行わない）

/Qopt-report-file:stderr は、旧レポートと同じく標準出力に表示

“:ファイル名” では、デフォルトのファイル名を変更できる

/Qopt-report-format:vs は、Visual Studio* IDE に適した形式で出力

デバッグビルド (-g ：Linux* と OS X*、 /Zi ： Windows*)では、アセンブリー・コードとオブジェクト・ファイルにループの最適化情報が含まれる

- /Qopt-report-embed は、リリースビルドでこの機能を有効にする

出力されるレポートのフィルター処理

- 最適化レポートは巨大になる可能性がある
- フィルター処理で、アプリケーションのパフォーマンスに重要な部分だけを抽出できる
- /Qopt-report-routine:<function1>[,<function2>,…]
  “function1” には、関数名の正規表現もしくは、その一部分を指定できる。また、特定の行番号の範囲を指定できる：

  icl /Qopt-report-filter : “test.cpp,100-300” test.cpp

- また、特定の最適化フェーズに注目できる：

  /Qopt-report-phase : [フェーズ]

ループのベクトル化と並列化フェーズ

- ループの入れ子を階層表示
  - 可読性を高め理解し易くする
  - コンパイラーが複数バージョンのループを生成した場合、それぞれ個別のメッセージが生成される
- コードがインライン展開されると、呼び出し側(caller)/呼ばれた側(callee) が明記される
- "loop" フェーズが指定されると、メッセージにはメモリーとキャッシュの最適化情報（ブロッキング、アンロール、プリフェッチなど）が含まれます
  - 統合されたベクトル化と並列化レポート

階層表示された
ループの最適化レポート

```
1  double a[1000][1000],b[1000][1000],c[1000][1000];
2
3  void foo() {
4  int i,j,k;
5
6    for( i=0; i<1000; i++) {
7      for( j=0; j< 1000; j++) {
8        c[j][i] = 0.0;
9        for( k=0; k<1000; k++) {
10         c[j][i] = c[j][i] + a[k][i] * b[j][k];
11       }
12     }
13   }
14 }
```

ソースの場所

ヘッダー情報

```
LOOP BEGIN at ...¥mydir¥dev¥test.c(7,5)
  Distributed chunk2
  ....
 LOOP BEGIN at ...¥mydir¥dev¥test.c(9,7)
   Distributed chunk2
   ....
 LOOP BEGIN at ...¥mydir¥dev¥test.c(6,3)
   ....
 LOOP END
 LOOP BEGIN at ...¥mydir¥dev¥test.c(6,3)
   ....
 LOOP END
LOOP END
LOOP END
```

レポートの内容

ループの入れ子

14.0 と 15.0 & 16.0 のループ最適化レポート

```
HPO VECTORIZER REPORT (foo) LOG OPENED ON Mon Feb 24 12:10:15 2014
<d:¥dev¥test.c;-1:-1;hpo_vectorization;foo;0>
HPO Vectorizer Report (foo)
d:¥dev¥test.c(7:5-7:5):VEC:foo: loop was not vectorized:
loop was transformed to memset or memcpy
d:¥dev¥test.c(6:3-6:3):VEC:foo: PERMUTED LOOP WAS VECTORIZED
d:¥dev¥test.c(9:7-9:7):VEC:foo: loop was not vectorized: not inner loop
d:¥dev¥test.c(7:5-7:5):VEC:foo: loop was not vectorized: not inner loop
HLO REPORT LOG OPENED ON Mon Feb 24 12:10:15 2014
<test.c;-1:-1;hlo;foo;0>
High Level Optimizer Report (foo)
<d:¥dev¥test.c;7:7;hlo_distribution;in foo;0>
LOOP DISTRIBUTION in foo at line 7
<d:¥dev¥test.c;6:6;hlo_linear_trans;foo;0>
LOOP INTERCHANGE in loops at line: 6 7
Loopnest permutation ( 1 2 ) --> ( 2 1 )
LOOP INTERCHANGE in loops at line: 6 7 9
Loopnest permutation ( 1 2 3 ) --> ( 2 3 1 )
<d:¥dev¥test.c;7:7;hlo_reroll;foo;0>
Loop at line:7 memset generated
Block, Unroll, Jam Report:
(loop line numbers, unroll factors and type of transformation)
<d:¥dev¥test.c;6:6;hlo_unroll;foo;0>
Loop at line 6 completely unrolled by 8
Loop Collapsing Report:
<d:¥dev¥test.c;7:7;hlo_loop_collapsing;foo;0>
Loops at line:7 and line:6 collapsed
```

```
Report from: Loop nest, Vector & Auto-parallelization
optimizations [loop, vec, par]
LOOP BEGIN at d:¥dev¥test.c(7,5)
Distributed chunk1
  remark #25430: LOOP DISTRIBUTION (2 way)
  remark #25448: Loopnest Interchanged : ( 1 2 ) --> ( 2 1 )
  remark #25424: Collapsed with loop at line 6
  remark #25412: memset generated
  remark #15144: loop was not vectorized: loop was transformed
to memset or memcpy
LOOP END

LOOP BEGIN at d:¥dev¥test.c(7,5)
Distributed chunk2
  remark #25448: Loopnest Interchanged : ( 1 2 3 ) --> ( 2 3 1 )
  remark #15018: loop was not vectorized: not inner loop

LOOP BEGIN at d:¥dev¥test.c(9,7)
Distributed chunk2
  remark #15018: loop was not vectorized: not inner loop

  LOOP BEGIN at d:¥dev¥test.c(6,3)
    remark #15145: vectorization support: unroll factor set to 4
    remark #15003: PERMUTED LOOP WAS VECTORIZED
  LOOP END

  LOOP BEGIN at d:¥dev¥test.c(6,3)
    remark #15003: REMAINDER LOOP WAS VECTORIZED
  LOOP END
LOOP END
LOOP END
```

ループ変換の順序付きレポート

複数バージョンのループ：
ピールループ、リマインダー・ループおよびコア部分

```
LOOP BEGIN at ggFineSpectrum.cc(124,5) inlined into ggFineSpectrum.cc(56,7)
  remark #15018: loop was not vectorized: not inner loop
  LOOP BEGIN at ggFineSpectrum.cc(138,5) inlined into ggFineSpectrum.cc(60,15)
  Peeled
    remark #25460: Loop was not optimized
  LOOP END
  LOOP BEGIN at ggFineSpectrum.cc(138,5) inlined into ggFineSpectrum.cc(60,15)
    remark #15145: vectorization support: unroll factor set to 4
    remark #15002: LOOP WAS VECTORIZED
  LOOP END
  LOOP BEGIN at ggFineSpectrum.cc(138,5) inlined into ggFineSpectrum.cc(60,15)
  Remainder
    remark #15003: REMAINDER LOOP WAS VECTORIZED
  LOOP END
LOOP END
```

ピールとリマインダーによるベクトル化

アセンブリー・リストへの注釈

```
.L11:    # optimization report
         # LOOP WAS INTERCHANGED
         # loop was not vectorized: not inner loop
   xorl %edi, %edi #38.3
   movsd b.279.0.2(%rax,%rsi,8), %xmm0 #41.32
   unpcklpd %xmm0, %xmm0 #41.32
         # LOE rax rcx rbx rsi rdi r12 r13 r14 r15 edx xmm0
..B1.11:          # Preds ..B1.11 ..B1.10
..L12: # optimization report
         # LOOP WAS INTERCHANGED
         # LOOP WAS VECTORIZED
         # VECTORIZATION HAS UNALIGNED MEMORY
REFERENCES
         # VECTORIZATION SPEEDUP COEFFECIENT 2.250000
   movaps a.279.0.2(%rcx,%rdi,8), %xmm1 #41.22
   movaps 16+a.279.0.2(%rcx,%rdi,8), %xmm2 #41.22
   movaps 32+a.279.0.2(%rcx,%rdi,8), %xmm3 #41.22
   movaps 48+a.279.0.2(%rcx,%rdi,8), %xmm4 #41.22
   mulpd %xmm0, %xmm1 #41.32
   mulpd %xmm0, %xmm2 #41.32
   <...>
```

```
L4::        ; optimization report
            ; PEELED LOOP FOR VECTORIZATION
$LN36:
$LN37:
   vaddss xmm1, xmm0, DWORD PTR [r8+r10*4]          ;4.5
snip snip snip
L5::        ; optimization report
            ; LOOP WAS VECTORIZED
            ; VECTORIZATION HAS UNALIGNED MEMORY REFERENCES
            ; VECTORIZATION SPEEDUP COEFFECIENT 8.398438
$LN46:
   vaddps ymm1, ymm0, YMMWORD PTR [r8+r9*4]         ;4.5
snip snip snip
L6::        ; optimization report
            ; LOOP WAS VECTORIZED
            ; REMAINDER LOOP FOR VECTORIATION
            ; VECTORIZATION HAS UNALIGNED MEMORY REFERENCES
            ; VECTORIZATION SPEEDUP COEFFECIENT 2.449219
$LN78:
   add r10, 4 ;3.3
snip snip snip
L7::        ; optimization report
            ; REMAINDER LOOP FOR VECTORIATION
$LN93:
   inc rax                                          ;3.3
```


Optimization Notice

ベクトル化のレポートレベル

/Qopt-report-phase:vec /Qopt-report:N

- N には、取得するレポートの詳細レベルを指定；
  N が省略された場合は N = 2 がデフォルト

- レベル 0：ベクトル化レポートなし
- レベル 1：ベクトル化が行われた場合をレポート
- レベル 2：レベル 1 に加え、ベクトル化が行われなかった場所と簡単な診断をレポート
- レベル 3：ループベクトル化の診断を追加
- レベル 4：データのアライメントなど詳細情報を追加
  レベル 5：依存性情報を追加

レポートへの対応例(1)

$ icc -c -opt-report=4 -opt-report-phase=loop,vec -opt-report-file=stderr foo.c

```
最適化レポート開始: foo(float *, float *)
  レポート: ループの入れ子、ベクトルの最適化 [loop, vec]

LOOP BEGIN at W:¥sample¥code¥foo.c(4,2)
<Multiversioned v1>
  remark #25228: データの依存関係のループをマルチバージョンにしました。
  remark #15389: ベクトル化のサポート: 参照 theta にアラインされていないアクセスが含まれています。
  remark #15389: ベクトル化のサポート: 参照 sth にアラインされていないアクセスが含まれています。
  remark #15381: ベクトル化のサポート: ループ本体内でアラインされていないアクセスが使用されました。
  remark #15399: ベクトル化のサポート: アンロールファクターが 2 に設定されます。
  remark #15417: ベクトル化のサポート: 浮動小数点数をアップコンバートします (単精度から倍精度 1)。
  remark #15418: ベクトル化のサポート: 浮動小数点数をダウンコンバートします (倍精度から単精度 1)。
  remark #15300: ループがベクトル化されました。
  remark #15450: マスクなし非アライン・ユニット・ストライド・ロード: 1
  remark #15451: マスクなし非アライン・ユニット・ストライド・ストア: 1
  remark #15475: --- ベクトルループのコストサマリー開始 ---
  ....
  remark #15488: --- ベクトルループのコストサマリー終了 ---
  remark #25015: ループの最大トリップカウントの予測=32
LOOP END
LOOP BEGIN at W:¥sample¥code¥foo.c(4,2)
<Multiversioned v2>
  remark #15304: ループ はベクトル化されませんでした: マルチバージョンのベクトル化できないループ・インスタンスです。
LOOP END
```

```
#include <math.h>
void foo (float * theta, float *
sth) {
   int i;
   for (i = 0; i < 128; i++)
      sth[i] =
sin(theta[i]+3.1415927);
}
```

レポートへの対応:例(2)

$ icc -c -opt-report=4 -opt-report-phase=loop,vec -opt-report-file=stderr -fargument-noalias foo.c

(Windows* では、 **/Qalias-args-**)

```
最適化レポート開始: foo(float *, float *)

  レポート: ループの入れ子、ベクトルの最適化 [loop, vec]

LOOP BEGIN at W:¥sample¥code¥foo.c(4,2)
  remark #15389: ベクトル化のサポート: 参照 theta にアラインされていないアクセスが含まれています。
  remark #15389: ベクトル化のサポート: 参照 sth にアラインされていないアクセスが含まれています。
  remark #15381: ベクトル化のサポート: ループ本体内でアラインされていないアクセスが使用されました。
  remark #15399: ベクトル化のサポート: アンロールファクターが 2 に設定されます。
  remark #15417: ベクトル化のサポート: 浮動小数点数をアップコンバートします (単精度から倍精度 1)。
  remark #15418: ベクトル化のサポート: 浮動小数点数をダウンコンバートします (倍精度から単精度 1)。
  remark #15300: ループがベクトル化されました。
  remark #15450: マスクなし非アライン・ユニット・ストライド・ロード: 1
  remark #15451: マスクなし非アライン・ユニット・ストライド・ストア: 1
  remark #15475: --- ベクトルループのコストサマリー開始 ---
  remark #15476: スカラーループのコスト: 114
  remark #15477: ベクトルループのコスト: 40.750
  remark #15478: スピードアップの期待値: 2.790
  remark #15479: 低オーバーヘッドのベクトル操作: 9
  remark #15480: 中オーバーヘッドのベクトル操作: 1
  remark #15481: 高オーバーヘッドのベクトル操作: 1
  remark #15482: ベクトル化された算術ライブラリーの呼び出し: 1
  remark #15487: 型変換: 2
  remark #15488: --- ベクトルループのコストサマリー終了 ---
  remark #25015: ループの最大トリップカウントの予測=32
LOOP END
```

```
#include <math.h>
void foo (float * theta, float * sth) {
  int i;
  for (i = 0; i < 128; i++)
    sth[i] = sin(theta[i]+3.1415927);
}
```

レポートへの対応:例(3)

$ icc -c -opt-report=4 -opt-report-phase=loop,vec -opt-report-file=stderr -fargument-noalias foo2.c

```
最適化レポート開始: foo(float *, float *)

  レポート: ループの入れ子、ベクトルの最適化 [loop, vec]

LOOP BEGIN W:¥sample¥code¥foo2.c(4,2)
  remark #15389: ベクトル化のサポート: 参照 theta にアラインされていないアクセスが含まれています。
  remark #15389: ベクトル化のサポート: 参照 sth にアラインされていないアクセスが含まれています。
  remark #15381: ベクトル化のサポート: ループ本体内でアラインされていないアクセスが使用されました。
  remark #15300: ループがベクトル化されました。
  remark #15450: マスクなし非アライン・ユニット・ストライド・ロード: 1
  remark #15451: マスクなし非アライン・ユニット・ストライド・ストア: 1
  remark #15475: --- ベクトルループのコストサマリー開始 ---
  remark #15476: スカラーループのコスト: 111
  remark #15477: ベクトルループのコスト: 20.500
  remark #15478: スピードアップの期待値: 5.400
  remark #15479: 低オーバーヘッドのベクトル操作: 8
  remark #15481: 高オーバーヘッドのベクトル操作: 1
  remark #15482: ベクトル化された算術ライブラリーの呼び出し: 1
  remark #15488: --- ベクトルループのコストサマリー終了 ---
  remark #25015: ループの最大トリップカウントの予測=32
LOOP END
```

```
#include <math.h>
void foo (float * theta, float * sth) {
  int i;
  for (i = 0; i < 128; i++)
    sth[i] = sinf(theta[i]+3.1415927f);
}
```

レポートへの対応:例(4)

$ icc -c -opt-report=4 -opt-report-phase=loop,vec -opt-report-file=stderr -fargument-noalias -xavx foo2.c

最適化レポート開始: foo(float *, float *)

レポート: ループの入れ子、ベクトルの最適化 [loop, vec]

LOOP BEGIN at W:¥sample¥code¥foo2.c(4,2)
remark #15389: ベクトル化のサポート: **参照 theta にアラインされていないアクセスが含まれています。**
remark #15389: ベクトル化のサポート: **参照 sth にアラインされていないアクセスが含まれています。**
remark #15381: ベクトル化のサポート: **ループ本体内でアラインされていないアクセスが使用されました。**
remark #15300: ループがベクトル化されました。
remark #15450: **マスクなし非アライン・ユニット・ストライド・ロード: 1**
remark #15451: **マスクなし非アライン・ユニット・ストライド・ストア: 1**
remark #15475: --- ベクトルループのコストサマリー開始 ---
remark #15476: スカラーループのコスト: 110
remark #15477: ベクトルループのコスト: 11.620
remark #15478: **スピードアップの期待値: 9.410**
remark #15479: 低オーバーヘッドのベクトル操作: 8
remark #15481: 高オーバーヘッドのベクトル操作: 1
remark #15482: ベクトル化された算術ライブラリーの呼び出し: 1
remark #15488: --- ベクトルループのコストサマリー終了 ---
remark #25015: **ループの最大トリップカウントの予測=16**
LOOP END

```
#include <math.h>
void foo (float * theta, float * sth) {
  int i;
  for (i = 0; i < 128; i++)
    sth[i] = sinf(theta[i]+3.1415927f);
}
```

レポートへの対応:例(5)

$ icc -c -opt-report=4 -opt-report-phase=loop,vec -opt-report-file=stderr -fargument-noalias -xavx foo3.c

最適化レポート開始: foo(float *, float *)

レポート: ループの入れ子、ベクトルの最適化 [loop, vec]

LOOP BEGIN W:¥sample¥code¥foo3.c(6,2)
remark #15388: **ベクトル化のサポート: 参照 theta にアラインされたアクセスが含まれています。**
remark #15388: **ベクトル化のサポート: 参照 sth にアラインされたアクセスが含まれています。**
remark #15300: ループがベクトル化されました。
remark #15448: **マスクなしアライン・ユニット・ストライド・ロード: 1**
remark #15449: **マスクなしアライン・ユニット・ストライド・ストア: 1**
remark #15475: --- ベクトルループのコストサマリー開始 ---
remark #15476: スカラーループのコスト: 110
remark #15477: ベクトルループのコスト: 9.870
remark #15478: **スピードアップの期待値: 11.130**
remark #15479: 低オーバーヘッドのベクトル操作: 8
remark #15481: 高オーバーヘッドのベクトル操作: 1
remark #15482: ベクトル化された算術ライブラリーの呼び出し: 1
remark #15488: --- ベクトルループのコストサマリー終了 ---
remark #25015: **ループの最大トリップカウントの予測=16**
LOOP END

```
#include <math.h>
void foo (float * theta, float * sth) {
  int i;
  __assume_aligned(theta,32);
  __assume_aligned(sth,32);
  for (i = 0; i < 128; i++)
    sth[i] = sinf(theta[i]+3.1415927f);
}
```

レポートへの対応:例(6)

$ icc -c -opt-report=4 -opt-report-phase=loop,vec -opt-report-file=stderr -fargument-noalias -xavx foo4.c

最適化レポート開始: foo(float *, float *)

レポート: ループの入れ子、ベクトルの最適化 [loop, vec]

LOOP BEGIN W:¥sample¥code¥foo4.c(6,2)
remark #15388: ベクトル化のサポート: 参照 theta にアラインされたアクセスが含まれています。
remark #15388: ベクトル化のサポート: 参照 sth にアラインされたアクセスが含まれています。
remark #15412: ベクトル化のサポート: sth のストリーミング・ストアが生成されました。
remark #15300: ループがベクトル化されました。
remark #15448: マスクなしアライン・ユニット・ストライド・ロード: 1
remark #15449: マスクなしアライン・ユニット・ストライド・ストア: 1
remark #15467: **マスクなしアライン・ストリーミング・ストア: 1**
remark #15475: --- ベクトルループのコストサマリー開始 ---
remark #15476: スカラーループのコスト: 110
remark #15477: ベクトルループのコスト: 9.870
remark #15478: スピードアップの期待値: 11.130
remark #15479: 低オーバーヘッドのベクトル操作: 8
remark #15481: 高オーバーヘッドのベクトル操作: 1
remark #15482: ベクトル化された算術ライブラリーの呼び出し: 1
remark #15488: --- ベクトルループのコストサマリー終了 ---
remark #25015: **ループの最大トリップカウントの予測=250000**
LOOP END

```
#include <math.h>
void foo (float * theta, float * sth) {
  int i;
  __assume_aligned(theta,32);
  __assume_aligned(sth,32);
  for (i = 0; i < 2000000; i++)
    sth[i] = sinf(theta[i]+3.1415927f);
}
```

IPO レポートの例

```
1       #include <stdio.h>
2
3       static void __attribute__((noinline)) bar
(float a[100][100], float b[100][100]) {
4       int i, j;
5       for (i = 0; i < 100; i++) {
6       for (j = 0; j < 100; j++) {
7       a[i][j] = a[i][j] + 2 * i;
8               b[i][j] = b[i][j] + 4 * j; 9          }
10      }
11      }
12
13      static void foo(float a[100][100],
float b[100][100]) {
14      int i, j;
15      for (i = 0; i < 100; i++) {
16      for (j = 0; j < 100; j++) {
17      a[i][j] = 2 * i;
18      b[i][j] = 4 * j;
19      }
20      }
21      bar(a, b);
22      }
23
24      extern int main() {
25      int i, j;
26       float a[100][100];
27        float b[100][100];
28
29      for (i = 0; i < 100; i++) {
30              for (j = 0; j < 100; j++) {
31                  a[i][j] = i + j;
32                   b[i][j] = i - j; 33
                }
34  }
35  foo(a, b);
36  foo(a, b);
37  fprintf(stderr, "%d %d¥n",
                                a[99][9], b[1]99]);
38      }
```

コンパイル:
icc -opt-report=3 –opt-report-phase=ipo sm.c

IPO レポートの機能 - インライン展開
-opt-report-phase=ipo –opt-report=3

インライン展開を制御するオプション値

ソースファイル sm.c の main 関数の行番号 24 に関するレポート

foo() は、行番号 35 と 36 でインライン展開された

bar() は、foo の行番号 21 で呼び出されているが、main にインライン展開されていない

外部関数 fprintf

行番号 3 のユーザー定義関数 bar() は、関数呼び出しを持たない

行番号 13 のスタティック関数 foo() は、インライン展開されるため実行されない

インライン展開オプション値:
-inline-factor: 100
…
インライン展開レポート: (main) [1] sm.c(24,19)
-> INLINE: [35] foo()
-> [21] bar()
-> INLINE: [36] foo()
-> [21] bar()
->EXTERN: [37] fprintf
インライン展開レポート: (bar) [2] sm(3,81)
実行されないスタティック関数: (foo) sm.c(13,55)

IPO レポートの機能 - さらに詳しく
-opt-report-phase=ipo –opt-report=4

プログラム全体の最適化レポート

全ルーチンの % がコンパイルされた

sz = 中間言語単位でそれぞれインライン可能なルーチンのサイズ (total = (stmts + exprs))

isz = インライン展開による呼び出しもとのサイズ増加

インライン展開されなかった理由

プログラム全体 (SAFE) [いずれかのメソッド]: true
プログラム全体 (SEEN) [テーブルメソッド]: true
プログラム全体 (READ) [オブジェクト・リーダー・メソッド]: false

インライン展開レポート: (main) [1/3=33.3%]
sm.c(24,19)
-> INLINE: [35] foo (isz = 40) (sz = 47 (25+22))
-> [21] bar() (isz = 47) (sz = 54 (24+30))
[[呼び出し先は noinline でマークされています]]
-> INLINE: [35] foo (isz = 40) (sz = 47 (25+22))
-> [21] bar() (isz = 47) (sz = 54 (24+30))
[[呼び出し先は noinline でマークされています]]
-> EXTERN: [37] fprintf

オフロード・レポート
インテル® Xeon Phi™ コプロセッサー用

- –opt-report-phase=offload でコンパイル
- ホストとコプロセッサー向けの個別のレポートを生成
- インテル® Cilk™ Plus キーワードと OpenMP* 4.0 プラグマ/宣言子を使用したオフロードのレポート

- OpenMP* 4.0 オフロード・プラグマの例：

  icc –c -openmp -opt-report-phase=offload offload_test.c

オフロード・レポート – OpenMP を使用した例

```
1.  #pragma omp declare target
2.  int compute(int i) { return i++; }
3.  #pragma omp end declare target
4.  int do_offload() {
5.      int i = 0;
6.  #pragma omp target map(tofrom:i)
7.          { i = compute(i); }
8.      return i;
9.  }
```

ホストレポート

offload_test.c(6-6):OFFLOAD:do_offload: Offload to target MIC 1

Data sent from host to target
i, scalar size 4 bytes
Data received by host from target
i, scalar size 4 bytes

コプロセッサー・レポート

offload_test.c(6-6):OFFLOAD:do_offload: Outlined offload region

Data received by target from host
i, scalar size 4 bytes
Data sent from target to host
i, scalar size 4 bytes

その他の最適化フェーズのレポート

-opt-report-phase=

par	自動並列化レポート、ベクトル化レポートと同じ構成
openmp	OpenMP 構文に関するレポート。ループレポートにマージ
pgo	プロファイルに基づく最適化のレポート。有用なプロファイルを持つ関数情報を含む
cg	コード生成時の最適化レポート。組み込み関数の置き換えなど
loop	ループとメモリー最適化のレポート。キャッシュ・ブロッキング、プリフェッチ、スカラー置換など
tcollect	インテル® Trace Analyzer 向けのデータ収集に関するレポート

旧オプションと新オプションの対応

-vec-report、-par-report および -openmp-report は、廃止予定。バージョン 14.x コンパイラーと同じ出力とはならない

代わりに、もっとも近いフェーズと新しい最適化レポート・レベルが出力される。–opt-report-file=stderr が指定されないと標準出力には出力されない。必要であれば設定ファイルにこのオプションを加えると良いでしょう

新しいオプションを利用することを推奨。makefile の "clean" セクションに *.optrpt ファイルの削除を追加すると良いでしょう

Microsoft* Visual Studio* でのレポート

15.0 & 16.0 コンパイラーでは、統合開発環境（IDE)上で新しいレポートを参照できる。次のオプションを使用：

/Qopt-report-format:vs

プロジェクト・プロパティーの [診断 (インテル(R) コンパイラー）] にある [最適化診断] セクションと、 [ツール] メニューの [オプション]->[インテル(R) コンパイラーおよびツール] にある [最適化レポート] で設定

テキスト・エディター中の最適化レポートをクリックして、caller もしくは callee を調査できる

レポート･ツール・ウィンドウが表示され、ソース行を直接ダブルクリックして、最適化レポートをフィルター処理

コンパイラー最適化サポート・ツール・ウインドウ

レポートを階層表示

ダブルクリックでソース位置にジャンプ

クリックで呼び出し先にジャンプ

Compiler Optimization Report

0 VEC | 0 PAR | 0 OpenMP | 83 PGO | 38 LNO | 0 Offload | 0 CG

Search Optimization Report

Open Documents
Current Project
Current Document
Current Function
Current Selection

		Inline Info	File	Line	Column	Project
7	ar>::~_Yarn<char>: profile feedback used a st		..\..\..\..\Progr	3387	3	Sample
8	char_t>::~_Yarn<wchar_t>: profile feedback us		..\..\..\..\Progr	3387	3	Sample
8	eedback used a static estimate profile		Sample.cpp	29	1	Sample
8	eedback used a static estimate profile		Sample.cpp	42	1	Sample
8	le feedback used a static estimate profile		Sample.cpp	68	1	Sample
8		Sample.cpp...	Sample.cpp	58	3	Sample
	25430: LOOP DISTRIBUTION (2 way) (line: 57, column: 2)					
	25448: Loopnest Interchanged : (1 2) --> (2 1) (line: 57, column: 2)					
	25424: Collapsed with loop at line 57 (line: 57, column: 2)					
	25412: memset generated (line: 57, column: 2)					
	15144: loop was not vectorized: loop was transformed to memset or memcpy (line: 58, column: 3)					
85	Main loop	Sample.cpp...	Sample.cpp	57	2	Sample
86	Remainder	Sample.cpp...	Sample.cpp	57	2	Sample
87	Distributed chunk2	Sample.cpp...	Sample.cpp	60	4	Sample
88	Distributed chunk2	Sample.cpp...	Sample.cpp	58	3	Sample
89	Main loop	Sample.cpp...	Sample.cpp	11	2	Sample
90	Remainder					
91	Main loop					
92	Remainder					
93	Main loop					

Output | Error List | Compiler Optimization Report | Compiler Inline Report

コンテキスト・スコープでメッセージをフィルター

最適化フェーズでメッセージをフィルター

キーワードでメッセージをフィルター

関連情報

インテル® Parallel Studio XE 2016 Composer Edition
コンパイラー・ユーザーおよびリファレンス・ガイド インテル・プレミア・サポート https://premier.intel.com

法務上の注意書きと最適化に関する注意事項

本資料に掲載されている情報は現状のまま提供され、いかなる保証もいたしません。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスを許諾するためのものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品性に関する保証、第三者の特許権、著作権、その他、知的財産権の侵害への保証を含む) をするものではありません。

性能に関するテストに使用されるソフトウェアとワークロードは、性能がインテル® マイクロプロセッサー用に最適化されていることがあります。SYSmark* や MobileMark* などの性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。製品の購入を検討される場合は、他の製品と組み合わせた場合の本製品の性能など、ほかの情報や性能テストも参考にして、パフォーマンスを総合的に評価することをお勧めします。

最適化に関する注意事項

インテル® コンパイラーは、互換マイクロプロセッサー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化が行われない可能性があります。これには、
インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2)、インテル® ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。インテル® マイクロアーキテクチャーに非固有の特定の最適化は、インテル製マイクロプロセッサー向けに予約されています。この注意事項の適用対象である特定の命令セットの詳細は、該当する製品のユーザー・リファレンス・ガイドを参照してください。

改訂 #20110804

Optimization Notice