コードの現代化(最適化) 1-2-3

インテル® コンパイラーを使用する際に直面するよくある問題と課題

2017 年 7 月

iSUS 編集長

すがわら きよふみ

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

このセッションの内容

- ベクトル化と命令セット
- ベクトル化を支援するコンパイラーの機能
- インテル® Advisor を使用した最適化のステップ
- 複数ソケットシステム (NUMA) 環境での留意点

このセッションで使用する機材:
インテル® Core™ i7-4790K プロセッサー 4.00GHz (メモリー帯域幅 25.6GB/s) 、インテル® Xeon® プロセッサー E5-2699 v4 2.20GHz

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化と命令セット

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ソフトウェアに対するハードウェアの影響
コア数、→スレッド数、→ ベクトル幅

* ark.intel.com で公開されている出荷済の製品仕様

	インテル® Xeon® プロセッサー 64 ビット	インテル® Xeon® プロセッサー 5100 番台	インテル® Xeon® プロセッサー 5500 番台	インテル® Xeon® プロセッサー 5600 番台	インテル® Xeon® プロセッサー 開発コード名 Sandy Bridge EP	インテル® Xeon® プロセッサー 開発コード名 Ivy Bridge EP	インテル® Xeon® プロセッサー 開発コード名 Haswell EP	インテル® Xeon® プロセッサー 開発コード名 Broadwell EP	インテル® Core™ X プロセッサー 開発コード名 Skylake-X	インテル® Xeon Phi™ コプロセッサー 開発コード名 Knights Corner	インテル® Xeon Phi™ プロセッサー & コプロセッサー 開発コード Knights Landing
コア	1	2	4	6	8	12	18	22	6 ~ 18	61	72
スレッド	2	2	8	12	16	24	36	44	12 ~ 36	244	288
SIMD 幅	128	128	128	128	256	256	256	256	512	512	512
	インテル® SSE3 (2004)	インテル® SSE4.1 (2006)	インテル® SSE4.2 (2009)	(2010)	インテル® AVX (2011)	(2013)	インテル® AVX2 (2014)	(2016)	インテル® AVX512 (2017)	IMIC	インテル® AVX512

ハイパフォーマンス･ソフトウェアは両方を活用する必要がある:

- 並列性 (マルチスレッド、マルチプロセス) とベクトル化 (SIMD)

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

命令セットが拡張されるときに何が起こるのか？

- SIMD レジスターの幅が同じ場合 (SSE2 -> SSE4.2)
  - 新しい命令や機能をすぐに利用できます
- SIMD レジスターの幅が異なる場合 (SSE4.2 -> AVX)
  - コンパイラーやプロセッサーが利用できても、OS が新しい命令セットをサポートするのを待つ必要があります

… ただし …

SIMD 幅にスケーラブルなコードを作成しておきましょう

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

インテル® コンパイラーのベクトル化オプション

自動ベクトル化

- /Qax<SIMD 命令セット,…> (-ax<SIMD,…>)
- /Qx<SIMD 命令セット> (-x<SIMD>)
- /QxHOST (-xHOST)

SIMD 命令セットに指定可能なキーワード:

SSE2, SSE3, SSSE3, SSE4.1, SSE4.2,	(128 ビット)
ATOM_SSSE3, ATOM_SSE4.2,	(128 ビット)
AVX, CORE-AVX-I, CORE-AVX2,	(256 ビット)
MIC-AVX512, CORE-AVX512, COMMON-AVX512	(512 ビット)

/O2 (-O2) 以上が指定されると自動ベクトル化が有効となる:
これらのオプションが指定されない場合のデフォルト命令セットは SSE2

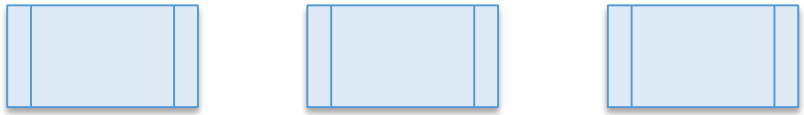
© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化オプションはパフォーマンスに最も影響する
/Qx (-x) と /Qax (-ax)

- すべてのソースファイルに同じオプションを指定することを推奨

  … しかし困難な場合がある …

- オブジェクト･ファイルが提供される、事情によりオプションが制限される



main.c　　func1.c　　func2.obj (または特定のオプションが指示される)

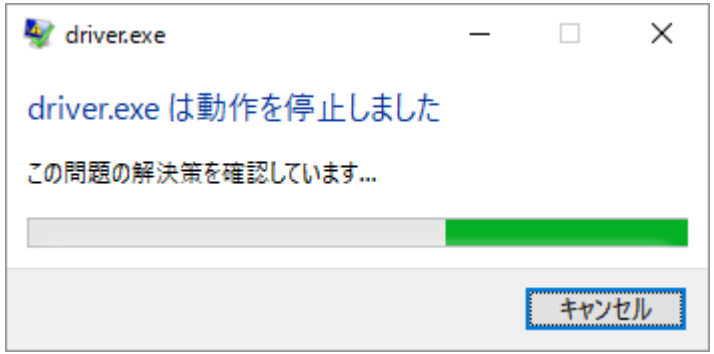
> icl main.c func1.c func2.obj /QaxCORE-AVX2 /Fetest_code.exe
> または
> icl func2.c /QxCORE-AVX512 /c
> icl main.c fuinc1.c func2.obj /QaxCORE-AVX2 /Fetest_code.exe

実行すると?

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化オプションはパフォーマンスに最も影響する (2)
/Qx (-x) と /Qax (-ax)

- 問題となる: ターゲットの実行環境が最新のプロセッサーではない、
  インテル® AVX/AVX2 からインテル® AVX-512 への移行中のコードが含まれる…



必ずしも例外が発生するとは限らない ……

ベクトル化できなければ
インテル® AVX-512 命令は
生成されない

コンパイラーのレポートやインテル® Advisor を使用して、
ベクトル化されているかどうかを確認

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化オプションはパフォーマンスに最も影響する (3)

ランタイムチェックの影響

```
C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>icl multiply.c /c /QaxCORE-AVX512 /QxCO
RE-AVX2 /Qopenmp /nologo
multiply.c

C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>icl driver.c multiply.obj /QaxCORE-AVX2
 /nologo
driver.c

C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>driver

ROW:101 COL: 101
Execution time is 3.269 seconds
GigaFlops = 6.241052
Sum of result = 195853.999899
```

```
C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>icl multiply.c /c /QxCORE-AVX2 /Qopenmp
 /nologo
multiply.c

C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>icl driver.c multiply.obj /QaxCORE-AVX2
 /nologo
driver.c

C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>driver

ROW:101 COL: 101
Execution time is 2.872 seconds
GigaFlops = 7.103760
Sum of result = 195853.999899
```

> icl multiply.c /c /QaxCORE-AVX512 /QxCORE-AVX2
> icl driver.c multiply.obj /QaxCORE-AVX2 /Fedriver.exe

> icl multiply.c /c /QxCORE-AVX2
> icl driver.c multiply.obj /QaxCORE-AVX2 /Fedriver.exe

処理中に数万回、数十万回呼び出されるような下位の関数では、
ランタイムチェックを行うべきではない

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化オプションはパフォーマンスに最も影響する (4)

バイナリーに適用されている命令セットは?

Program metrics

Elapsed Time 2.96s

Vector Instruction Set AVX

Number of CPU Threads 1

Loop metrics

Total CPU time	2.88s	100.0%
Time in 1 vectorized loop	2.87s	99.7%
Time in scalar code	0.01s	

Vectorization Gain/Efficiency

Vectorized Loops Gain/Efficiency	1.80x	~45%
Program Approximate Gain	1.80x	

ベクトル命令セットに
インテル® AVX が適用されている

サーベイレポートで
詳細を確認

ROOFLINE

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	Why No Vectorization?	Vectorized Loops: Vect...	Vectorized Loops: Efficiency	Vectorized Loops: Gain...	Vectorized Loops: VL (...	Trip Counts	Instruction Set Analysis: Traits	Instruction Set Analysis: Data T...	Instruction Set Analysis: Num..	Advanced	Location
[loop in matvec at multiply.c:15]		2.799s	2.799s 97.2%	Inside vectorized						101	Inserts	Float64	5		multiply.c:15
[loop in matvec at multiply.c:15]		0.040s	0.040s	Scalar						101		Float64	3		multiply.c:15
[loop in matvec at multiply.c:12]	1 Ineffective ...	0.030s	2.869s	Vectorized (Bo...		AVX	44%	1.77x	4	25; 1		Float64	3; 6		multiply.c:12
[loop in matvec at multiply.c:12]		0.030s	2.829s 98.3%	Vectorized (Bo...		AVX			4	25		Float64	6		multiply.c:12
[loop in matvec at multiply.c:12]		0.000s	0.040s	Remainder						1		Float64	3		multiply.c:12
matvec		0.010s	2.879s	Function							Inserts	Float64	6		multiply.c:8
_scrt_common_main_seh		0.000s	2.879s	Function									0		exe_common.inl:201
[loop in main at driver.c:98]		0.000s	2.879s	Scalar	loop with function ...					1000000		Float64	2		driver.c:98
main		0.000s	2.879s	Function							Extracts; Type Convers...	Float32...	14		driver.c:26

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

コード中で CPU の命令セットをチェックする

```
#include <immintrin.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
  const unsigned long knl_features =
      (_FEATURE_AVX512F | _FEATURE_AVX512ER |
       _FEATURE_AVX512PF | _FEATURE_AVX512CD );
  if ( _may_i_use_cpu_feature( knl_features ) )
    printf("This CPU supports AVX-512F+CD+ER+PF as introduced in Knights Landing¥n");
  else
    printf("This CPU does not support all Knights Landing AVX-512 features¥n");
  return 1;
}
```

インテル® コンパイラーは、プロセッサーの機能をチェックする
_may_i_use_cpu_feature() 組込み関数をサポート

https://www.isus.jp/specials/target-intel-core-processors/
https://www.isus.jp/products/c-compilers/how-to-detect-knl-instruction-support/

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化を支援する
コンパイラーの機能

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化に影響する 6 つの要因

ループ伝搬依存

```
DO I = 1, N
   A(I+1) = A(I) + B(I)
ENDDO
```

関数呼び出し

```
for (i = 1; i < nx; i++) {
  x = x0 + i * h;

     sumx = sumx + func(x, y, xp);
}
```

ポインター･エイリアシング

```
void scale(int *a, int *b){
   for (int i = 0; i < 1000; i++)
         b[i] = z * a[i];
}
```

不明なループカウント

```
struct _x { int d; int bound; };

void doit(int *a, struct _x *x)
{
  for(int i = 0; i < x->bound; i++)
    a[i] = 0;
}
```

間接メモリーアクセス

```
 for (i=0; i<N; i++)
      A[B[i]] = C[i]*D[i]
```

外部ループ

```
for(i = 0; i <= MAX; i++) {
  for(j = 0; j <= MAX; j++) {
    D[i][j] += 1;
  }
}
```

さらに ....

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化を支援するコンパイラー・オプション
/Qipo (-ipo)

- プロシージャー間の最適化 (IPO) は、マルチステップの自動処理で、コンパイラーがコードを解析してどの最適化が有効であるかを判断できるようにします

```
double a[512];
main(){
…
func (&a[0], &a[256], ct);
…
}
```

source1.c

```
func(double *p,
      double *q int ct){
…
  for(i=0; i<ct; i++)
    p[i] += q[i] * z;
…
}
```

source2.c

インライン展開およびその他の最適化は、プロファイル情報によって向上

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

IPO による最適化

アドレスの解析
配列次元のパディング
エイリアス解析
配列の自動転置
メモリープールの自動生成
C++ クラス階層解析
共通ブロック変数の統合
共通ブロックの分割
定数の伝播
不要な呼び出しの検出
不要な仮引数の排除
不要な関数の排除
仮引数のアライメント解析
前方代入
間接呼び出し変換

インライン展開
mod/ref 解析
不要な呼び出しの部分的に排除
レジスターに引数を渡して呼び出しとレジスターの使用を最適化
ポインター解析
ルーチンのキー属性の伝播
専用化
スタックフレームのアライメント
構造体分割とフィールドの並べ替え
シンボル･テーブル･データの促進
未参照変数の削除
プログラム全体の解析

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化を支援するコンパイラーオプション

/Qipo (-ipo) 使用時の注意事項

- IPO を使用して各ソースファイルがコンパイルされるたびに、コンパイラーはソースコードの中間表現 (IR) を擬似オブジェクト・ファイルに格納する。擬似オブジェクト・ファイルには、通常のオブジェクト・ファイルの代わりに IR が含まれる
- コンパイラーはリンカーの直前に起動され、コンパイラーは、すべての擬似オブジェクト・ファイルを対象に IPO を実行する (.o や .obj にオブジェクトは含まれない)
  ipo を適用しないオブジェクトと、ipo を適用したオブジェクトをリンクする場合は、インテルのリンクツールを使用

ライブラリーを作成する場合やリンク時は、lib (ar) や link (ld) の代わりに
専用ツール、xilib (xiar) と xilink (xild) を使用する

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ipo を適用しないオブジェクトと ipo を適用したオブジェクト

Intel Compiler 17.0 Update 2 Intel(R) 64 Visual Studio 2015

```
> icl multiply.c /c /nologo
multiply.c
multiply.c(11): 警告 #3180: 識別できない OpenMP* プラグマです。
  #pragma omp simd
          ^



> icl driver.c multiply.obj /Qipo /nologo
driver.c
ipo: 警告 #11003: オブジェクト・ファイル multiply.obj に IR がありません。このフ
ァイルは -Qipo で コンパイルされていません。

>
```


© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化を支援するコンパイラーの機能

言語機能 (1)

言語機能	説明
__declspec(align(*n*))	変数を *n* バイト境界にアライメントするようにコンパイラーに指示する。変数のアドレスは *address* mod n=0
__declspec(align(*n*,off))	各 *n* バイト境界内でオフセットだけ離して変数を *n* バイト境界にアライメントするようにコンパイラーに指示する。変数のアドレスは *address* mod n=off
__assume_aligned(*a*,*n*)	配列 *a* が *n* バイト境界にアライメントされていると見なすようにコンパイラーに指示する。アライメント情報が取得できなかった場合に使用する
__assume(*cond*)	このキーワードが指定されている場合は、cond が true であると仮定するようにコンパイラーに指示する。通常、より効率的なコードを生成するため、アライメント情報のような利用可能な特性をコンパイラーに伝える
#pragma ivdep	ベクトル依存性が存在していると推定されてもそれを無視するようにコンパイラーに指示
#pragma novector	ループをベクトル化しないように指定

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

iSUS

ベクトル化を支援するコンパイラーの機能

言語機能 (2)

- ソースコードに複数の関数が含まれている場合、アルゴリズムやメモリー参照の方法により、特定の SIMD 命令セットで最適な性能を発揮するような状況を想定してください

```
func1 () {….}
func2 () {….}
func3 () {….}
```

source_code.c

インテル® SSE4.2 で最高の性能を発揮!!

> icl source_code.c /QxCORE-AVX-I

#pragma [intel] optimization_parameter target_arch=SSE4.2
を関数宣言の直前に追加すると、対象の関数だけ命令セットを変更可能

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化を支援するコンパイラーの機能

/Qvec- (-no-vec) オプション

- ベクトル化によりアプリケーションがどれくらい恩恵を得られているか簡単に調査
- /Qvec- (-no-vec) オプションが指定されるとコンパイラーは、SIMD 命令を使用するが、ベクトル化せずにスカラー操作を行うコードを生成
- /Qvec- (-no-vec) ありと、なしのバイナリーを作成してパフォーマンスを比較

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

インテル® SSE2

Intel Compiler 17.0 Update 2 Intel(R) 64 Visual Studio 2015

```
> icl driver.c multiply.c /Qipo /nologo
driver.c
multiply.c

> driver

ROW:101 COL: 101
Execution time is 2.436 seconds
GigaFlops = 8.375205
Sum of result = 195853.999899

> icl driver.c multiply.c /Qipo /nologo /Qvec-
driver.c
multiply.c

> driver

ROW:101 COL: 101
Execution time is 6.534 seconds
GigaFlops = 3.122436
Sum of result = 195853.999899
```

インテル® AVX2

Intel Compiler 17.0 Update 2 Intel(R) 64 Visual Studio 2015

```
> icl driver.c multiply.c /Qipo /nologo /QxCORE-AVX2
driver.c
multiply.c

> driver

ROW:101 COL: 101
Execution time is 1.935 seconds
GigaFlops = 10.543669
Sum of result = 195853.999899

> icl driver.c multiply.c /Qipo /nologo /QxCORE-AVX2 /Qvec-
driver.c
multiply.c

> driver

ROW:101 COL: 101
Execution time is 5.052 seconds
GigaFlops = 4.038401
Sum of result = 195853.999899
```

デフォルトの状態を検証

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化を支援するコンパイラーの機能

ベクトル化レポート

- /Qopt-report (-qopt-report) を指定すると、コンパイラーが適用した、または適用できなかった最適化に関するレポートを出力

```
レポート: ループの入れ子、ベクトル、自動並列化の最適化 [loop, vec, par]
LOOP BEGIN at C:¥code¥matmul¥multiply.c(13,2)
  リマーク #15541: 外部 loop は自動ベクトル化されませんでした: SIMD ディレクティブの使用を検討してください。

  LOOP BEGIN at C:¥code¥matmul¥multiply.c(15,3)
    リマーク #15344: loop はベクトル化されませんでした: ベクトル依存関係がベクトル化を妨げています。
    最初の依存関係を以下に示します。詳細については、レベル 5 のレポートを使用してください。
    リマーク #15346: ベクトル依存関係: FLOW の依存関係が b[i] (16:4) と b[i] (16:4) の間に仮定されました。
    リマーク #25439: 剰余ありアンロール - 2
  LOOP END

  LOOP BEGIN at C:¥code¥matmul¥multiply.c(15,3)
  <Remainder>
  LOOP END
LOOP END
```

このレポートは、Visual Studio® 内で利用可能

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

ベクトル化のレポートレベル

/Qopt-report:*N* (-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、ベクトル化さ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

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

コンパイラー・レポートのオプション

- /Qopt-report-phase:フェーズ[,フェーズ,…]
  フェーズ = loop, par, vec, openmp, ipo, pgo, cg, offload, tcollect, all
- /Qopt-report-file:stdout | stderr | ファイル名
- /Qopt-report-embed
- /Qopt-report-routine:<関数名>[,<関数名>]
- /Qopt-report-filter: “ソースファイル名, 行番号-行番号”

メッセージを英語にする:
/Qdiag-message-catalog- (Windows*)
export LANG=en_US.UTF-8 (Linux*)

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

インテル® Advisor を使用した最適化ステップ

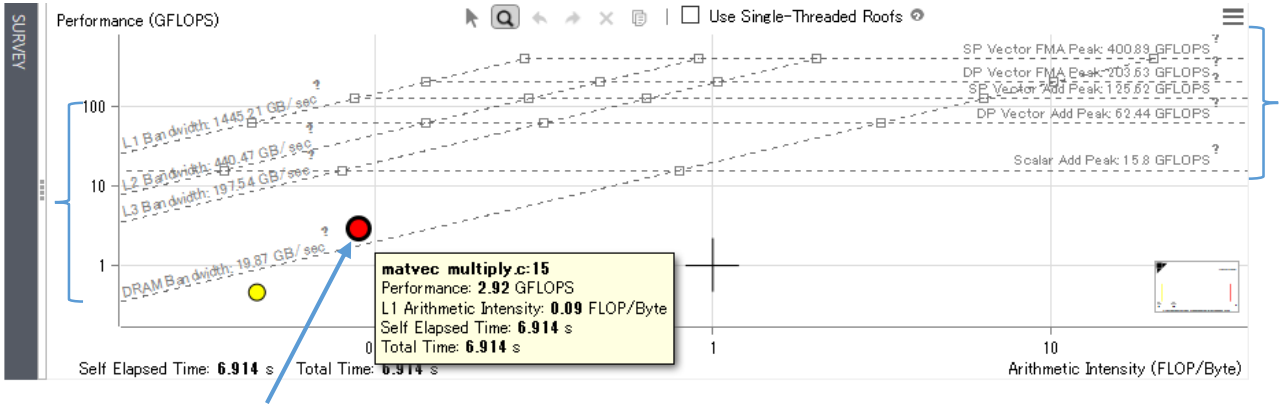
- 現状のコードの状況を知る
- 最適化の方針を決定する
- 最適化を適用する
- 最適化の効果を確認する

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

アプリケーションのパフォーマンスを知る新しい方法

インテル® Advisor の Roofline (ルーフライン) 解析を使用して、デフォルトコードのパフォーマンスを理解する: icl driver.c multiply.c /O2 /Zi

キャッシュとメモリーの潜在的なピーク性能



命令セットの潜在的なピーク性能

この段階でアプリケーションは、メモリー集約型であり SIMD 命令のロード/ストアデータはキャッシュに収まっていないか、活用していない

キャッシュを考慮したルーフライン解析のトレーニング: https://www.isus.jp/products/advisor/roofline_webinar/

このセッションで使用する機材: インテル® Core™ i7-4790K プロセッサー 4.00GHz (メモリー帯域幅 25.6GB/s)

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

サーベイ解析でさらに詳しく調査

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	FLOPS: GFLOPS	FLOPS: AI	Trip Counts: Average	Trip Counts: Min	Trip Counts: Max	Trip Counts: Call Count	Trip Counts: Iteration D...	Loop Instance Total Time	Why No Ve...	Instruction Set Analysis: Traits	Instruction Set Analysis: Data T...	Instruction Set Analysis: Num...
[loop in matvec at multiply.c:15]	2 Assumed d...	6.914s	6.914s	Scalar	2.922	0.089	50	50	50	101000000	< 0.001s	< 0.001s	vect...		Float64	3
Scalar loop. Not vectorized: vector dependence prevents vectorization Loop was unrolled by 2																
[loop in matvec at multiply.c:13]	1 Opportunit...	0.439s	7.354s 99.9%	Scalar	0.460	0.045	101	101	101	1000000	< 0.001s	< 0.001s	out...		Float64	3
matvec		0.010s	7.363s							1000000		< 0.001s				3
_scrt_common_main_seh												63s				0
[loop in main at driver.c:98]												63s	loo		Float64	2
main												63s		ifts; Sh...	Float32...	16

- ベクトル化されていないスカラーループ
- 2 回アンロールされている
- ループ回数は 50 回
- 処理全体で 101000000 回呼び出されている

Source | Top Down | Code Analytics | Assembly | Recommendati

File: multiply.c:15 matvec

```
1    #include "Multiply.h"
2
3    #ifdef NOALIAS
4    void matvec(int size1, int size2, FTYPE a[][size2], FTYPE b[
5    #else
6    //#pragma optimization_parameter target_arch=CORE-AVX512
7    void matvec(int size1, int size2, FTYPE a[][size2], FTYPE b[
8    #endif
9    {
10       int i, j;
11
12   #pragma omp simd
13       for (i = 0; i < size1; i++) {
14           b[i] = 0;
15           for (j = 0;j < size2; j++) {
16               b[i] += a[i][j] * x[j];
17           }
18       }
19   }
```

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

Module: default.exe!0x140007510

	Address	Line	Assembly	Total Time	%	Self Time	%
body	0x140007510		**Block 1:**				
	0x140007510	16	movsxd r13, r12d				
	0x140007513	16	lea r14d, ptr [r12+r12*1]	0.152s		0.152s	
	0x140007517	16	shl r13, 0x4	0.030s		0.030s	
	0x14000751b	15	inc r12d	1.219s		1.219s	
	0x14000751e	16	movsxd r14, r14d				
	0x140007521	16	movsd xmm1, qword ptr [rbx+r13*1]	0.140s		0.140s	
	0x140007527	16	mulsd xmm1, qword ptr [rdi+r14*8]	0.072s		0.072s	
	0x14000752d	16	addsd xmm0, xmm1	1.077s		1.077s	
	0x140007531	16	movsd qword ptr [r9+rbp*8], xmm0	0.130s		0.130s	
	0x140007537	16	movsd xmm2, qword ptr [rbx+r13*1+0x8]	0.996s		0.996s	
	0x14000753e	16	mulsd xmm2, qword ptr [rdi+r14*8+0x8]				
	0x140007545	16	addsd xmm0, xmm2	0.181s		0.181s	
	0x140007549	16	movsd qword ptr [r9+rbp*8], xmm0	1.737s		1.737s	
	0x14000754f	15	cmp r12d, r10d	1.181s		1.181s	
	0x140007552	15	jb 0x140007510 <Block 1>				

- データ型は double
- 使用されている
  レジスターは 3 個

ボトルネックとなっているループ

、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
す。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

iSUS

インテル® Advisor による推奨

- [Recommendations] タブでインテル® Advisor からの最適化の推奨を得られる
- [Why No Vectorization?] で詳細を得るにはコンパイラーのレポートが必要

ROOFLINE

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	FLO GFL
[loop in matvec at multiply.c:15]	2 Assumed d...	6.914s	6.914s	Scalar	2.92
[loop in matvec at multiply.c:13]	1 Opportunit...	0.439s	7.354s 99.9%	Scalar	0.46
matvec		0.010s	7.363s	Function	
_scrt_common_main_seh		0.000s	7.363s	Function	
[loop in main at driver.c:98]		0.000s	7.363s	Scalar	
main		0.000s	7.363s	Fu	

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

All known issues with all possible recommendations: C++ / Fortran

Issue: Assumed dependency present

The compiler assumed there is an anti-dependency (Write after read – WAR) or a true dependency (Read after write – RAW) in the loop. Improve performance by investigating the assumption and handling a

Recommendation: Confirm dependency is real — Confidence: Need More

There is no confirmation that a real (proven) dependency is present in the loop. To confirm: Run a Dependencies analysis.

Issue: Potential underutilization of FMA instructions

Your current hardware supports the AVX2 instruction set architecture (ISA), which enables the use of fused multiply-add (FMA) instructions. Improve performance by utilizing FMA instructions.

Recommendation: Target the AVX2 ISA — Confidence: Low

Although static analysis presumes the loop may benefit from FMA instructions available with the AVX2 ISA, no AVX2-specific code executed for this loop. To fix: Use the `xCORE-AVX2` compiler option to gene `axCORE-AVX2` compiler option to enable multiple, feature-specific, auto-dispatch code generation, including AVX2.

Windows* OS	Linux* OS
/QxCORE-AVX2 or /QaxCORE-AVX2	-xCORE-AVX2 or -axCORE-AVX2

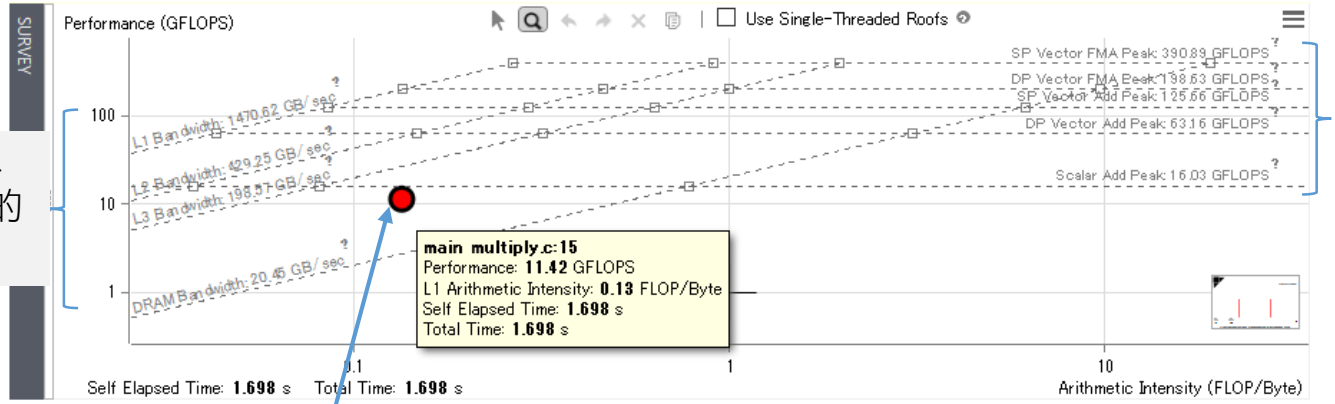
Read More:

- ax, Qax; x, Qx
- Code Generation Options in the Intel® C++ Compiler 16.0 User and Reference Guide
- Compiling for the Intel® Xeon Phi™ processor x200 and the Intel® AVX-512 ISA and Vectorization Resources for Intel® Advisor Users

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

-ipo でデフォルトのベクトル化を適用した結果

>icl driver.c multiply.c /Qipo /Zi /O2



キャッシュとメモリーの潜在的なピーク性能

命令セットの潜在的なピーク性能

この段階でアプリケーションの計算性能はかなり向上、L3 キャッシュを効率良く利用しつつある

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

サーベイ解析でさらに詳しく調査: ipo

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	FLOPS: GFLOPS	FLOPS: AI	Why No Vectorization?	Vectorized Loops: Vect...	Vectorized Loops: Efficiency	Vectorized Loops: Gain...	Vectorized Loops: VL (...	Trip Counts	Instruction Set Analysis: Traits	Instruction Set Analysis: Data T...	Instruction Set Analysis: Num..	Advanced
[loop in main at multiply.c:15]	1 Potential u...	1.698s	1.698s	Vectorized (Bo...	11.423	0.134		SSE2	91%	1.82x	2	12		Float32; F	5	Unrolled by 4
[loop in main at multiply.c:13]	1 Opportunit...	0.749s	2.447s	Scalar	2.291	0.380	inner loop was alre...					101	Unpacks	Float32...	12	
Scalar loop with instructions that use No loop transformations applied	Scalar loop with instructions that use SSE2 registers. Not vectorized: inner loop was already vectorized No loop transformations applied															
_scrt_common_main_seh		0.000s	2.447s	Function											0	

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

File: multiply.c:15 main

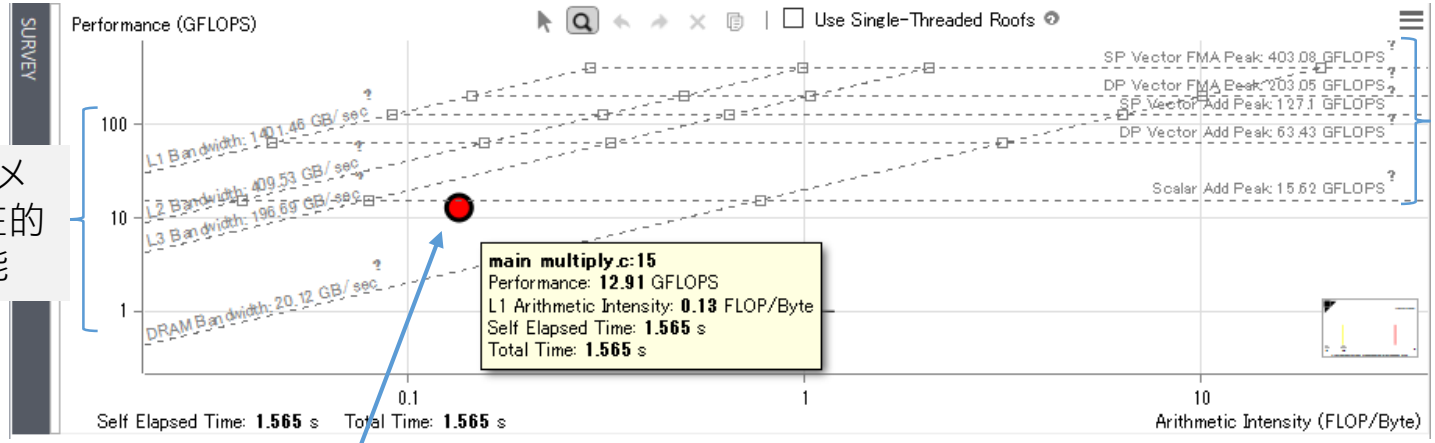
Line	Source	Total Time	%
1	#include "Multiply.h"		
2			
3	#ifdef NOALIAS		
4	void matvec(int size1, int size2, FTYPE a[][size2], FTYPE b[restrict], FTYPE x[])		
5	#else		
6	//#pragma optimization_parameter target_arch=CORE-AVX512		
7	void matvec(int size1, int size2, FTYPE a[][size2], FTYPE b[], FTYPE x[])		
8	#endif		
9	{		
10	int i, j;		
11			
12	#pragma omp simd		
13	for (i = 0; i < size1; i++) {		
14	b[i] = 0;		
15	for (j = 0;j < size2; j++) {	0.370s	
	[loop in main at multiply.c:15] Vectorized SSE; SSE2 loop processes Float32; Float64 data type(s) Loop was unrolled by 4		
16	b[i] += a[i][j] * x[j];	1.328s	
17	}		
18	}		
19	}		
20			

- インテル® SSE2 でベクトル化されている
- セルフ実行時間は、6.914 秒から 1.698 秒へ短縮
- 11.423 GLOPS
- ベクトル化効率 91% で、ベクトル長 2 であるためスカラーに対し 1.82 倍のゲイン
- 4 回のアンロールに加え、ベクトル化によりループ回数は 12 回に減少

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

インテル® AVX2 命令セットを適用後

>icl driver.c multiply.c /Qipo /Zi /O2 /QxCORE-AVX2



キャッシュとメモリーの潜在的なピーク性能

命令セットの潜在的なピーク性能

インテル® SSE2 でベクトル化されたルーフライン・レポートとほとんど変わらない

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

サーベイ解析でさらに詳しく調査: インテル® AVX2

ROOFLINE

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	FLOPS: GFLOPS	FLOPS: AI	Why No Vectorization?	Vectorized Loops: Vect...	Vectorized Loops: Efficiency	Vectorized Loops: Gain...	Vectorized Loops: VL (...	Trip Counts	Instruction Set Analysis: Traits	Instruction Set Analysis: Data T...	Instruction Set Analysis: Num...
[loop in main at multiply.c:15]		1.499s	1.499s	Vectorized (Bo...	13.479	0.134		AVX	77%	3.09x	4	25		Float64	3
[loop in main at multiply.c:13]	1 Opportunit...	0.420s	1.919s	Scalar	1.201	0.336	inner loop was alre...					101	FMA; Unpacks	Float64	5
_scrt_common_main_seh		0.000s	1.919s	Function											0
main		0.000s	1.919s	Function		0.040							Extracts; Unpacks	Float32...	7
[loop in main at driver.c:110]		0.000s	1.919s	Scalar		0.134	inner loop was alre...					1000000		Float64	7
[loop in main at driver.c:108]	1 Ineffective ...	n/a	n/a	Remainder Co...											

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

File: multiply.c:15 main

```
Line  Source
1     #include "Multiply.h"
2
3     #ifdef NOALIAS
4     void matvec(int size1, int size2, FTYPE a[][size2], FTYPE b[restrict], FTYPE x[])
5     #else
6     //#pragma optimization_parameter target_arch=CORE-AVX512
7     void matvec(int size1, int size2, FTYPE a[][size2], FTYPE b[], FTYPE x[])
8     #endif
9     {
10        int i, j;
11
12    #pragma omp simd
13        for (i = 0; i < size1; i++) {
14            b[i] = 0;
15            for (j = 0;j < size2; j++) {
              [loop in main at multiply.c:15]
                Vectorized AVX loop processes Float64 data type(s)
                No loop transformations applied
16                b[i] += a[i][j] * x[j];
17            }
18        }
19    }
20
```

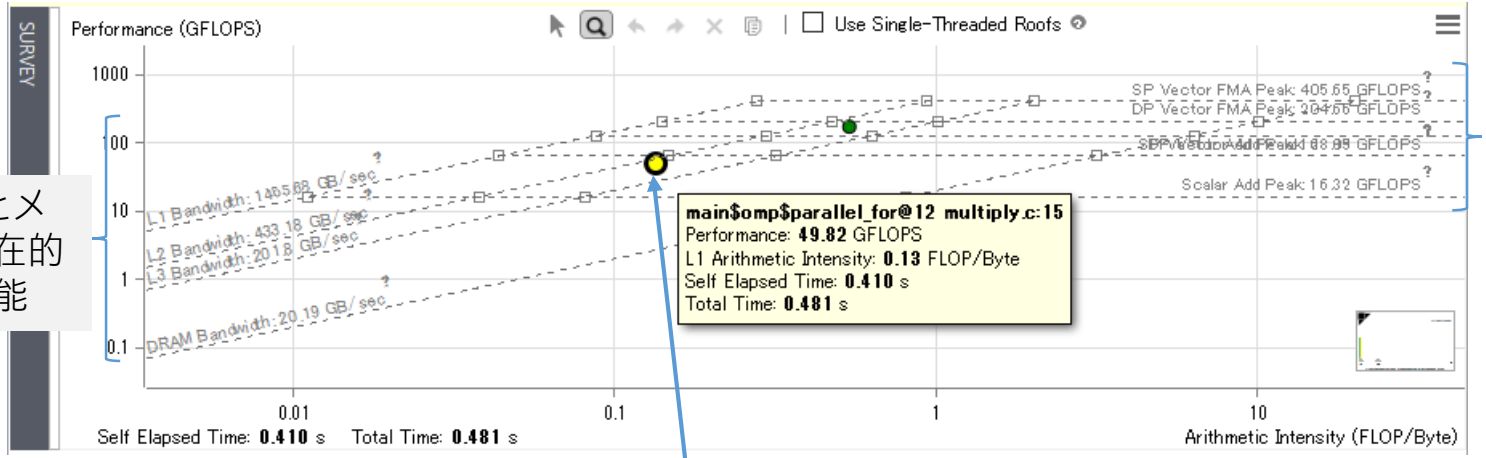
0.179s

- インテル® AVX でベクトル化されている (FMA)
- セルフ実行時間は、1.698 秒から 1.499 秒へ短縮
- 13.479 GLOPS
- ベクトル化効率 77% で、ベクトル長 4 であるためスカラーに対し 3.09 倍のゲイン
- アンロールは適用されず、ベクトル化によりループ回数は 25 回に減少

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

OpenMP* による並列化を適用後

>icl driver.c multiply.c /Qipo /Zi /O2 /QxCORE-AVX2 /Qopenmp



キャッシュとメモリーの潜在的なピーク性能

命令セットの潜在的なピーク性能

並列化によりキャッシュの利用率は向上し、パフォーマンスもかなり高まっている

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

サーベイ解析でさらに詳しく調査: OpenMP*

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	FLOPS GFLOPS	FLOPS AI	Why No Vectorization?	Vectorized Loops Vect...	Vectorized Loops Efficiency	Vectorized Loops Gain...	Vectorized Loops VL (...	Trip Counts	Instruction Set Analysis Traits	Instruction Set Analysis Data T...	Instruction Set Analysis Num...	Advanced
[loop in mainompparallel_for@12 at multi...	1 Ineffective ...	0.030s	0.030s	Vectorized (Bo...	1019.728	0.134		AVX2	42%	1.70x	4	6; 2; 3	FMA	Float64	3; 5	Unrolled by 4
[loop in mainompparallel_for@12 at m...		0.020s	0.020s	Vectorized (Bo...	1938.916	0.134		AVX2			4	6	FMA	Float64	5	Unrolled by 4
[loop in mainompparallel_for@12 at m...		0.010s	0.010s	Remainder	40.376	0.134						2		Float64	3	
[loop in mainompparallel_for@12 at m...		0.000s	0.000s	Peeled		0.134						3		Float64	3	
f _sert_common_main_sch		0.000s	23.881s	Function											0	

OpenMP* による並列化

```
7       void matvec(int size1, int size2, FTYPE a[][size2],                    x[])
8       #endif
9       {
10          int i, j;
11
12      #pragma omp parallel for private(j) num_threads(4)
13          for (i = 0; i < size1; i++) {
            [loop in main$omp$parallel_for@12 at multiply.c:13]
                Threaded (OpenMP) loop with instructions that use AVX registers. Not vectorized: inner loop was already vectorized
                Openmp loop
14              b[i] = 0;
15              for (j = 0;j < size2; j++) {
            [loop in main$omp$parallel_for@12 at multiply.c:15]
                Vectorized AVX; FMA loop processes Float64 data type(s) and includes FMA
                Loop was unrolled by 4
            [loop in main$omp$parallel_for@12 at multiply.c:15]
                Scalar remainder loop
                No loop transformations applied
            [loop in main$omp$parallel_for@12 at multiply.c:15]
                Scalar peeled loop
                No loop transformations applied
            [loop in main$omp$parallel_for@12 at multiply.c:15]
                Vectorized AVX remainder loop [not executed] processes Float64 data type(s)
                No loop transformations applied
16                  b[i] += a[i][j] * x[j];
17              }
18          }
19      }
```

Line	Total Time	%	Loop/Function Time	%	Traits
12	30.008ms				
15	10.006ms		30.008ms		
16	20.002ms				FMA

スレッド間でループが分割されたことで、スレッド内のループ回数は、6、2、または3。

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

インテル® AVX-512 命令を含むコード

>icl driver.c multiply.c /Qipo /Zi /O2 /QaxCORE-AVX512 /QxCORE-AVX2

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	FLOPS GFLOPS	FLOPS AI	Why No Vectorization?	Vectorized Loops Vector ISA	Efficiency	Gain...	VL (...	Trip Counts	Instruction Set Analysis Traits	Data Types
[loop in main at multiply.c:15]		1.679s	1.679s	Vectorized (Body)	12.032	0.134		AVX2			4	25	FMA	Float64
Vectorized AVX; FMA loop processes Flo No loop transformations applied		Vectorized AVX; FMA loop processes Float64 data type(s) and includes FMA No loop transformations applied												
[loop in main at multiply.c:15]		n/a	n/a	Vectorized (Body) [Not Executed]				AVX512			8		FMA	Float32; Float64
Vectorized AVX512F_512 loop [not execu Loop was unrolled by 4		Vectorized AVX512F_512 loop [not executed] processes Float32; Float64 data type(s) and includes FMA Loop was unrolled by 4												
[loop in main at multiply.c:13]	1 Potential u...	0.448s	2.127s	Scalar Versions	1.126	0.336	1 inner loop was alr...							

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

Function Call Sites and Loops						Vectorized Loops Vector ISA	VL (...	Com...	Instruction Set Analysis Traits	Data T...	Advanced	Location
Total												
RtlUserThreadStart												
BaseThreadInitThunk	100.0%	2.127s	0.000s	Function								
_scrt_common_main_seh	100.0%	2.127s	0.000s	Function								exe_common.inl:201
main	100.0%	2.127s	0.000s	Function					Extracts; FMA; Unpacks	Float32...		driver.c:51
[loop in main at driver.c:110]	100.0%	2.127s	0.000s	Scalar Versions	1 inner loop was alrea...					Float64		driver.c:110
[loop in main at driver.c:110]	100.0%	2.127s	0.000s	Scalar	inner loop was alread...					Float64		driver.c:110
[loop in main at multiply.c:13]	100.0%	2.127s	0.000s	Scalar Versions	1 inner loop was alrea...				FMA; Unpacks; [FMA;...	Float64...		multiply.c:13
[loop in main at multiply.c:13]	100.0%	2.127s	0.448s	Scalar	inner loop was alread...				FMA; Unpacks	Float64		multiply.c:13
[loop in main at multiply.c:15]	78.9%	1.679s	0.000s	Vectorized Versions		AVX2; [AVX512]	4; [8]	3.09x	FMA; [FMA]	Float64; .	[Unrolled by 4]	multiply.c:15
[loop in main at multiply.c:15]	78.9%	1.679s	1.679s	Vectorized (Body)		AVX2	4	3.09x	FMA	Float64		multiply.c:15
[loop in main at multiply.c:15]	n/a	n/a	n/a	Vectorized (Body) [Not Executed]		AVX512	8	5.05x	FMA	Float32; F	Unrolled by 4	multiply.c:15
[loop in main at driver.c:108]	n/a	n/a	n/a	Remainder Complete...								driver.c:108
									FMA; Permutes; Shuffles	Float32...		multiply.c:13

インテル® AVX-512 命令は実行されていない

インテル® AVX2 とインテル® AVX-512 命令で構成される

インテル® Advisor は実行されないコードパスのパフォーマンスを推測

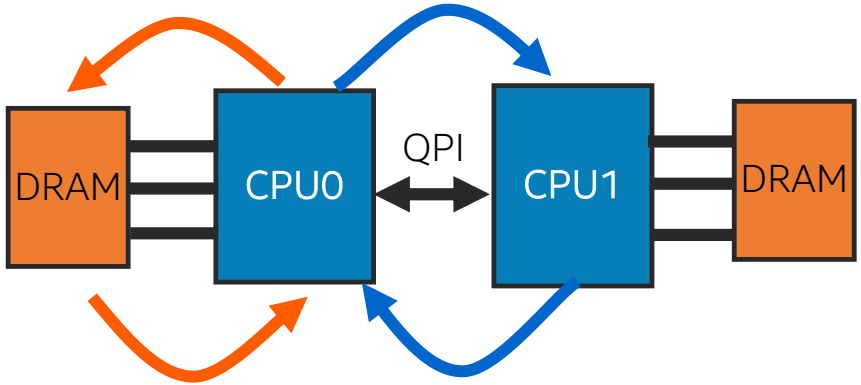
© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

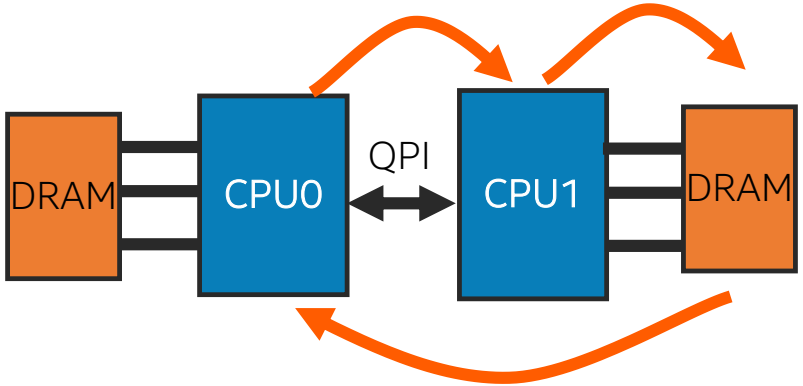
複数ソケットシステム (NUMA) 環境での留意点

あなたの開発したコードは、サーバー上で実行されているかもしれません …

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

NUMA 環境でのメモリーアクセス





ローカル・メモリー・アクセス

1. CPU0 は CPU0 側の DRAM にデータをリクエスト
2. CPU0 はデータが存在するか CPU1 に問い合わせ (snoop)
3. DRAM がデータを返す
4. CPU1 は問い合わせ (snoop) を返す

ローカルメモリーの待ち時間は、
このステップまでにかかる待ち時間

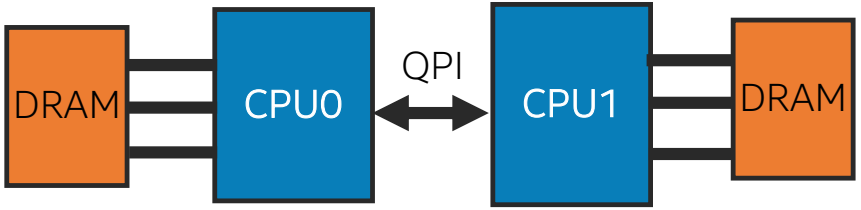
リモート・メモリー・アクセス

1. CPU0 は CPU1 にデータをリクエスト
2. リクエストに対し QPI を経由して CPU1 へ
3. CPU1 の IMC は (CPU1 側の) DRAM にリクエスト
4. CPU1 はキャッシュ内に問い合わせ (snoops)
5. データは QPI 経由で CPU0 へ返す

リモートメモリーの待ち時間 (レイテンシー) は、
待ち時間の短いインターコネクト次第

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

スレッドのアフィニティー制御と NUMA



NUMA 環境ではスレッドのアフィニティー (配置) がパフォーマンスに影響を及ぼす。

アフィニティーの制御:

- numactl コマンドを使用する
- KMP_AFFINITY を使用する

2 つのスレッドのアフィニティーを制御する例

参考文献:
https://www.isus.jp/products/vtune/pu27-03-identify-scalability-problem/
https://www.isus.jp/hpc/optimize-for-numa/

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

まとめ

- 最適化の第一歩は、ベクトル化に注目してください
- 効率良くベクトル化されたコードは、並列化による相乗効果を高めます
- 効率良くベクトル化および並列化されたコードは、将来のハードウェアへの準備を容易にします

- 開発ツールも日々進化しています
  - 最適化ツールを活用しましょう
  - 新しい機能を使ってみましょう

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

参考資料
コンパイラー・レポートの例

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

レポートへの対応例 (1)

$ icl /c /Qopt-report:4 /Qopt-report-phase:loop,vec /Qopt-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);
}
```


© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

レポートへの対応例 (2)

$ icl /c /Qopt-report:4 /Qopt-report-phase:loop,vec /Qopt-report-file:stderr /Qalias-args- foo.c

(Linux* では、**-fargument-noalias**)

```
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);
}
```


© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

レポートへの対応例 (3)

$ icl /c /Qopt-report:4 /Qopt-report-phase:loop,vec /Qopt-report-file:stderr /Qalias-args- 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);
}
```


© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

レポートへの対応例 (4)

$ icl /c /Qopt-report:4 /Qopt-report-phase:loop,vec /Qopt-report-file:stderr /Qalias-args- /Qxavx 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);
}
```


© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

レポートへの対応例 (5)

$ icl /c /Qopt-report:4 /Qopt-report-phase:loop,vec /Qopt-report-file:stderr /Qalias-args- /Qxavx 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);
}
```


© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

レポートへの対応例 (6)

$ icl /c /Qopt-report:4 /Qopt-report-phase:loop,vec /Qopt-report-file:stderr /Qalias-args- /Qxavx 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);
}
```


© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

iSUS の参考記事

- インテルのベクトル化ツール
  https://www.isus.jp/intel-vectorization-tools/
- インテル® Advisor 2017 ツールによるベクトル化のクイック解析
  https://www.isus.jp/products/advisor/quick-analysis-of-vectorization-using-advisor-2017-tool/
- 一般的なベクトル化のヒント
  https://www.isus.jp/products/c-compilers/common-vectorization-tips/
- インテル® AVX-512 で向上したベクトル化のパフォーマンス
  https://www.isus.jp/products/psxe/pu27-04-vectorization-opportunities/
- 自動ベクトル化が失敗した場合の対処
  https://www.isus.jp/products/c-compilers/what-to-do-when-auto-vectorization-fails/
- ベクトル化されたリダクション操作を記述できますか?
  https://www.isus.jp/products/c-compilers/can-you-write-vectorized-reduction-operation/

  https://www.isus.jp/products/c-compilers/vectorized-reduction-2-let-the-compiler-do/
- 間接呼び出しと仮想関数の呼び出し: インテル® C/C++ コンパイラー 17.0 によるベクトル化
  https://www.isus.jp/products/psxe/pu25-06-indirect-calls-and-virtual-functions-calls/
- ループをベクトル化するための条件
  https://www.isus.jp/products/psxe/requirements-for-vectorizable-loops/

© 2017 iSUS. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。*その他の社名、製品名などは、一般に各社の商標または登録商標です。

iSUS