2. 主要な最適化機能の活用方法

Revision 1.1

エクセルソフト株式会社
黒澤 一平

XLSOFT

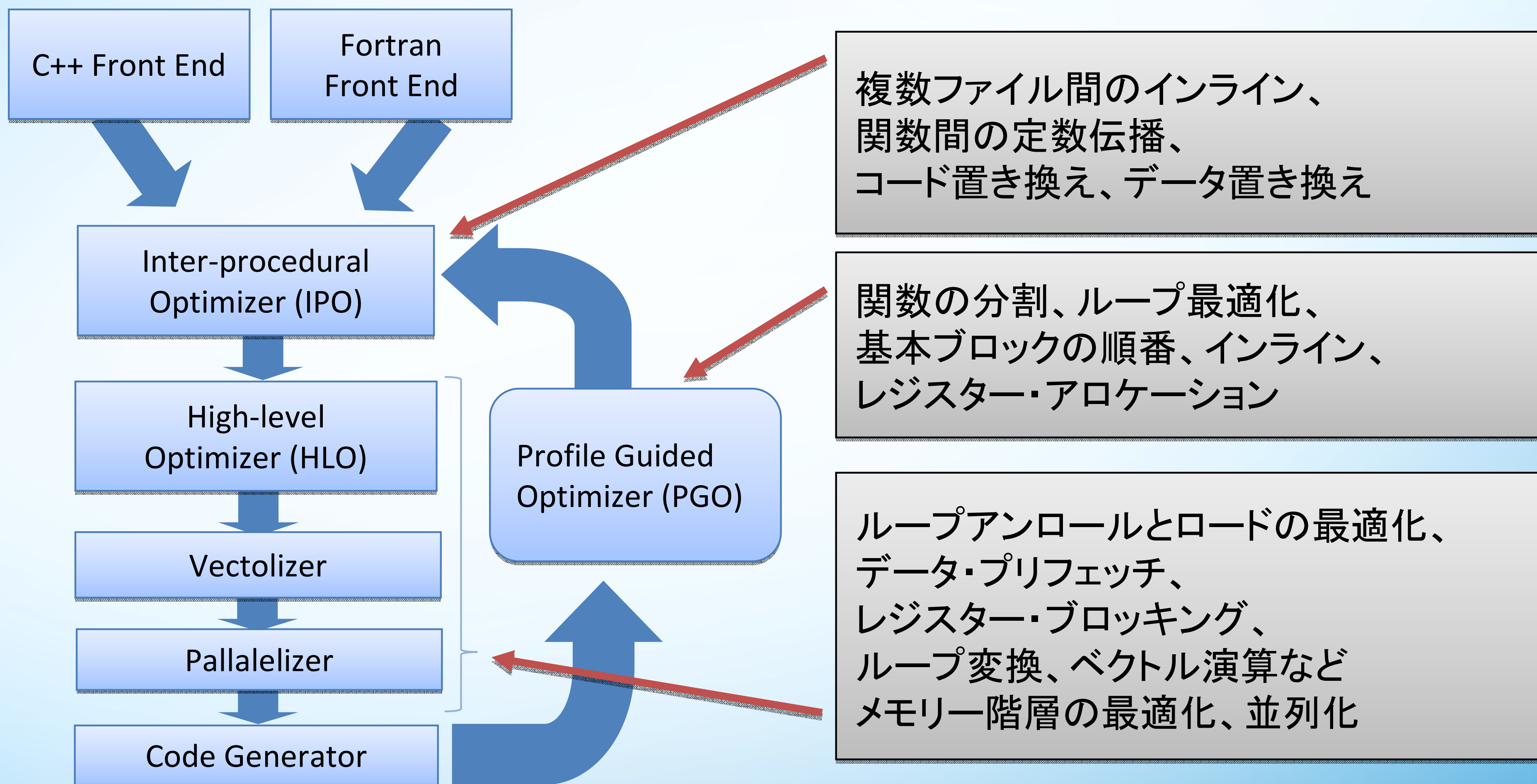
目次

- 効率的なベクトル化、最適化のためのコード記述方法
  - ベクトル化の仕様
  - キャッシュライン
  - 配列のアラインメント
  - 構造体のアラインメント
- プロシージャ間の最適化
- プロファイルに基づく最適化 (PGO)

XLSOFT

Copyright© 2011, XLsoft K.K. All rights reserved.

コンパイラーの最適化と構造



XLSOFT

Copyright© 2011, XLsoft K.K. All rights reserved.

High Performance Optimizer (HPO)
複数の最適化機能が複合的に適用

```
subroutine matmul(a,b,c,n)
real(8), dimension(n,n) :: a,b,c

c=0.d0                    ! 4
!$omp parallel do         ! 5
do i=1,n                  ! 6
  do j=1,n                ! 7
    do k=1,n              ! 8
      c(j,i)=c(j,i)+a(k,i)*b(j,k)
    enddo
  enddo
enddo
end
```

最適化レポート(簡略)

matmul.f90(5): (col. 7) : OpenMP DEFINED LOOP WAS PARALLELIZED
High Level Optimizer Report (matmul_)
matmul.f90(4): (col. 1) : LOOP WAS VECTORIZED.
matmul.f90(7): (col. 3) : PERMUTED LOOP WAS VECTORIZED.
LOOP INTERCHANGE in loops at line: 7 8
Loopnest permutation (1 2 3) --> (1 3 2)

Block, Unroll, Jam Report:
(loop line numbers, unroll factors and type of transformation)

Loop at line 7 blocked by 111
Loop at line 8 blocked by 111
Loop at line 6 blocked by 111

Loop at line 6 unrolled and jammed by 4
Loop at line 8 unrolled and jammed by 4

OpenMP による並列化、ベクトル化、ループ解析の最適化がすべて適用されている

XLSOFT
Copyright© 2011, XLsoft K.K. All rights reserved.

ループのベクトル化に必要な条件

- 独立性
  - できる限り、ループ反復の依存関係を排除する
- ループに関する制限:
  - ループ内の変数は明確でなければならない
  - 一部の依存ループはベクトル化が可能
  - インライン展開されていない関数はベクトル化できない
  - 一部の条件分岐はベクトル化を妨げる
  - ループはカウント可能でなければならない
  - ネストするループの外部ループはベクトル化できない
  - 混在データ型はベクトル化できない
  - その他

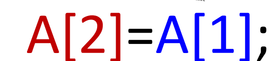
Copyright© 2011, XLsoft K.K. All rights reserved.

依存関係の例: フロー依存

- 書き込み後の読み取り(RAW)
- ループ間のフロー依存:
- 変数を書き込んでから
  別のループで読み取る

```
for (j=1; j<MAX; j++) {
        A[j]=A[j-1];
}

        A[1]=A[0];
        A[2]=A[1];
```



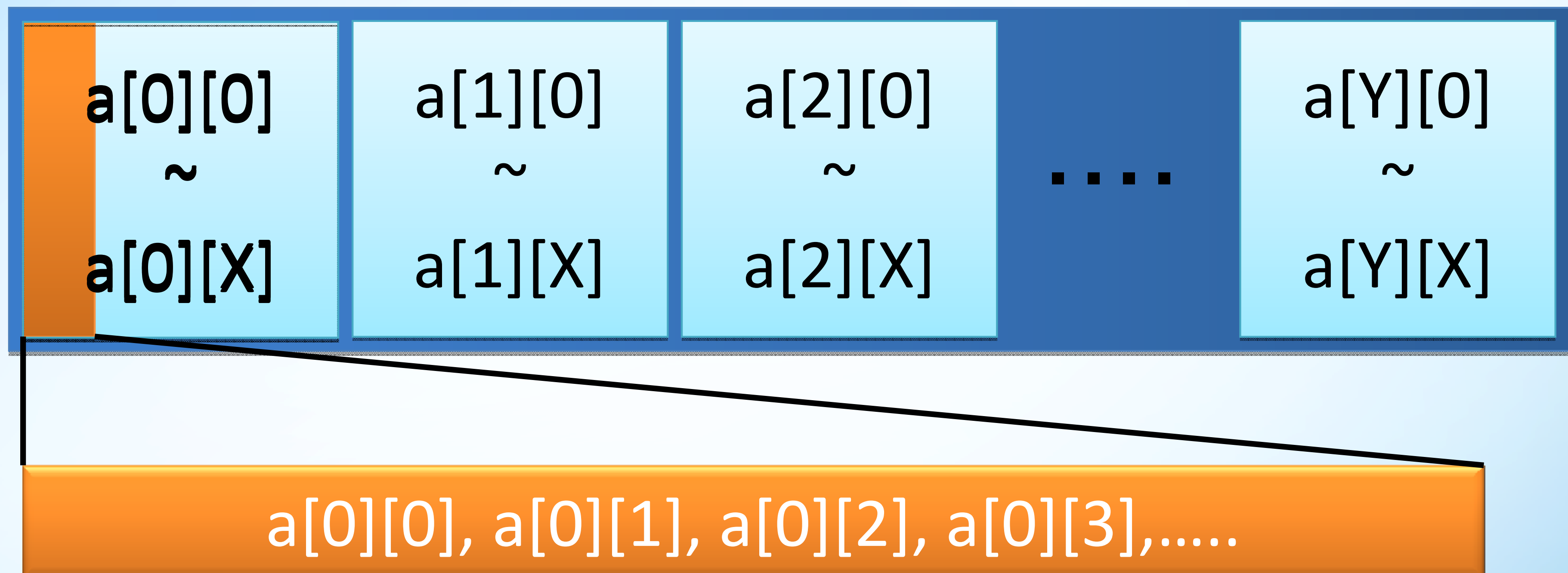
コンパイラーはアンロールやループ変換などを活用してベクトル化しようとする

XLSOFT

Copyright© 2011, XLsoft K.K. All rights reserved.

連続したメモリーアクセス

float a[Y+1][X+1];



ベクトル化においても、メモリー上に連続したデータ配置が有効

i が 1づつ増加する場合、
C/C++言語は　　a[j][i] が高速
Fortran 言語は a(i , j) が高速

Copyright© 2011, XLsoft K.K. All rights reserved.

ユニットストライドと非ユニットストライドの例

```
for (i=0;i<=MAX;i++)
    for (j=0;j<=MAX;j++) {
        c[i][j]+=1;   // ユニットストライド
        c[j][i]+=1;   // 非ユニットストライド
        A[j*j]+=1;    // 非ユニットストライド
        A[B[j]]+=1;   // 非ユニットストライド
        if (A[MAX-j]=1) last1=j;  // 非ユニットストライド
    }
```

- 非ユニットストライドはベクターデータの読み込み時間が演算時間を上回り、データ読み込み待ちが多発する

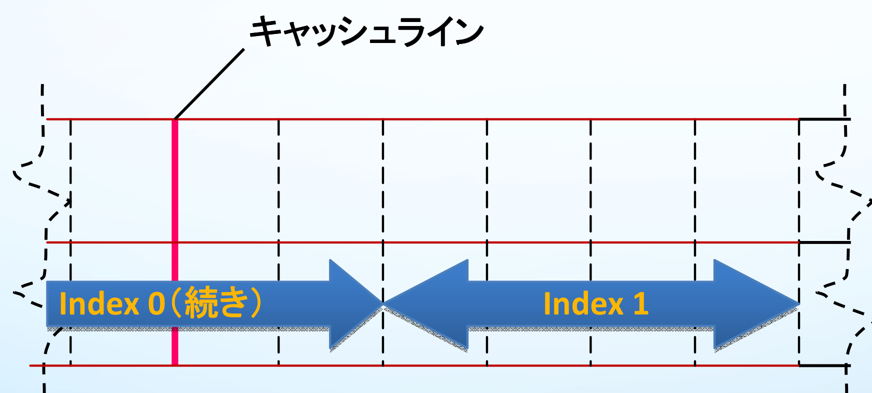
- コンパイラーは非効率な演算をベクトル化しない

XLSOFT

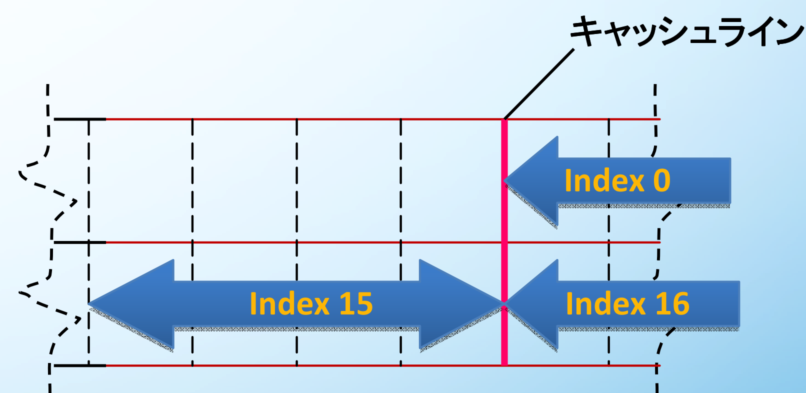
Copyright© 2011, XLsoft K.K. All rights reserved.

キャッシュラインの分割

- 定義: データ要素がキャッシュ境界をまたぐこと
- 影響: そのデータ要素にアクセスするには、1 回ではなく 2 回のメモリーアクセスが必要になるため、パフォーマンスが低下する
- キャッシュラインは 64 バイト



キャッシュライン分割の状態



キャッシュラインをまたがない状態

Copyright© 2011, XLsoft K.K. All rights reserved.

アラインメント

- movdqa 命令は、16バイト境界にアラインメントされているデータを高速転送することができる
- キャッシュラインの分割と同様に、高速演算のためには16バイト境界をまたがない配列の確保が有効
- AVX 使用時には 32バイト (256bit) のアラインメントが有効

```
__declspec(align(16)) float A[100][1024];
```

```
real*4 A(1024,100)
!DEC$ATTRIBUTES ALIGN: 16:: A
```


Copyright© 2011, XLsoft K.K. All rights reserved.

最適化のための配列の確保

- float または real*4 で 1250*1250 配列を確保する場合

```
float A[1250][1250];  キャッシュラインと 16,32バイト境界をまたぐ
real*4 A(1250,1250)   キャッシュラインと 16,32バイト境界をまたぐ
```

- 調整分(padding bytes)の 4バイト 14要素分(56バイト分) を追加
  4 * 1264 = 2 * 2 * 16 * 79

```
float A[1250][1264];
real*4 A(1264,1250)
```

多くの場合、ループ内で繰り返して使用される配列は、16(32)バイト境界のアラインメント、キャッシュラインを考慮した確保をすることが高速化につながる

XLSOFT

Copyright© 2011, XLsoft K.K. All rights reserved.

ベクトル化のために、配列がアラインされていることをコンパイラーに伝える

配列が宣言されているソースファイルと、使用されるソースファイルが異なる場合、コンパイラーに アラインされていることを伝えて最適化を補助する

C,C++

```
#pragma vector aligned
for (i=0; i<n; i++){
C[i] = A[i] * B[i];
}
```

Fortran

```
!DIR$ VECTOR ALIGNED
DO I=1, N
C(I) = A(I) * B(I)
ENDDO
```


Copyright© 2011, XLsoft K.K. All rights reserved.

構造体のアラインメントと、境界の指定

```
#pragma pack(push,thepack,1)
   struct PackedStr {
      char      Data;
      intptr_t Next;          /*Pointer */
      int       ExtraInfo;    /*int*/
   };
#pragma pack(pop,thepack)
```

または、すべての構造体をパックするコンパイラー・オプションを使用する

Linux*	Windows*
-Zp[n]	/Zp[n]

1、2、4、8、または 16 バイト境界で構造体をアラインする

XLSOFT

Copyright© 2011, XLsoft K.K. All rights reserved.

構造体のパディング方法

		x86	x86_64
Struct Astruct {			
char A; //		1	1
	//Pad	0	0
char B; //		1	1
	//Pad	2	6
void *B;//		4	8
	//Pad	0	0
char C; //		1	1
	//Pad	3	7
}	//Total	12 Byte	24 Byte

コンパイラーは自動的にパディングバイトを含め、構造体を最適化する

Copyright© 2011, XLsoft K.K. All rights reserved.

目次

- 効率的なベクトル化、最適化のためのコード記述方法
  - ベクトル化の仕様
  - キャッシュライン
  - 配列のアラインメント
  - 構造体のアラインメント
- プロシージャ間の最適化
- プロファイルに基づく最適化 (PGO)

XLSOFT

Copyright© 2011, XLsoft K.K. All rights reserved.

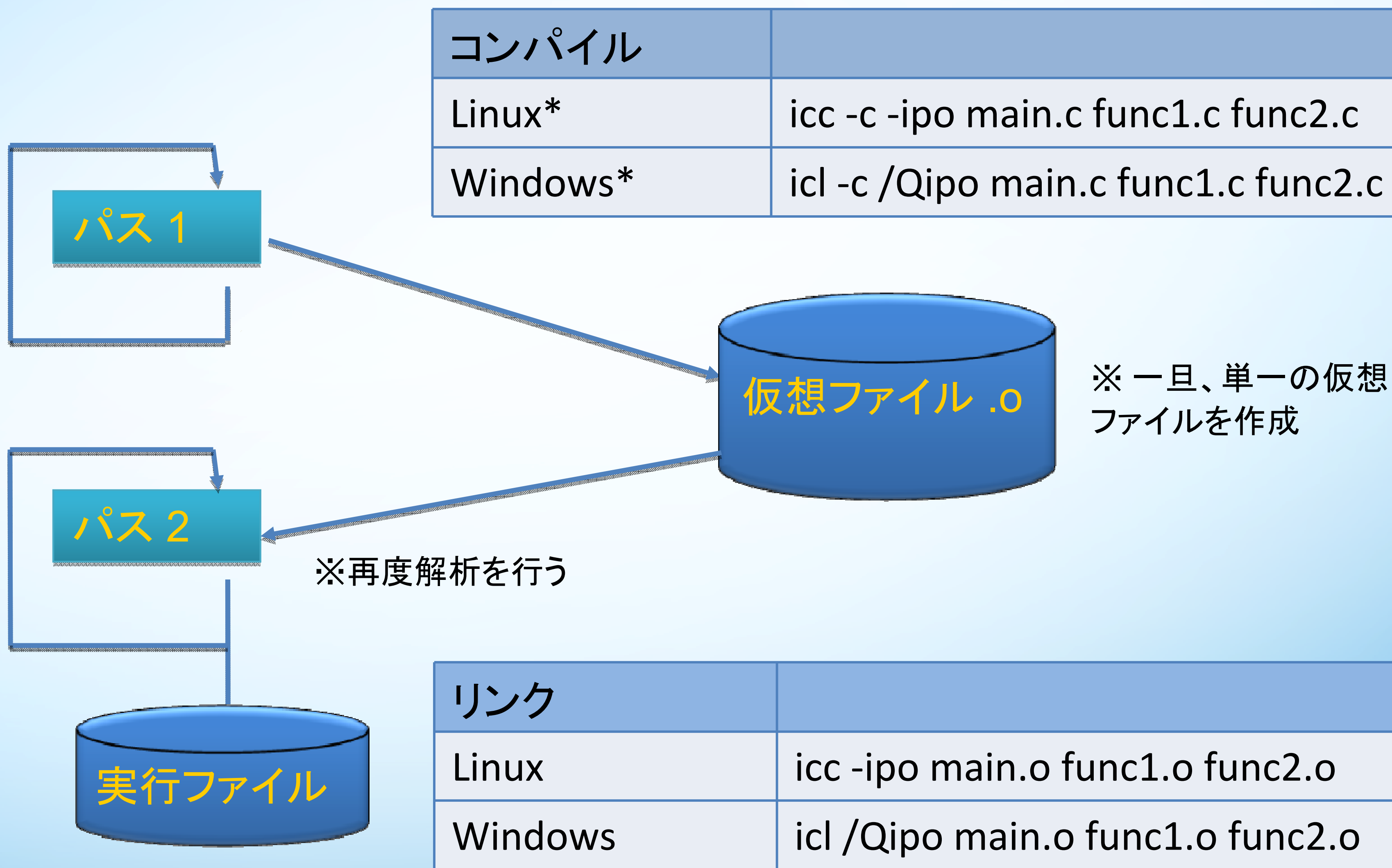
プロシージャ間の最適化

-ip:	単一ファイルのコンパイルでプロシージャー間の最適化を有効にする
-ipo:	複数ファイルにわたるプロシージャー間の最適化を有効にする

- 他のオプションと併用することでその最適化を支援する
- インライン展開以外に多くの利点を提供
  - プロシージャー間での定数伝播
  - レジスターでの引数の受け渡し
  - ループ不変コードの移動
  - 不要コードの排除
  - 部分的なインライン展開
  - ベクトル化やメモリーの明確化を支援

Copyright© 2011, XLsoft K.K. All rights reserved.

プロシージャ間の最適化 - IPO
2 ステップによる処理



コンパイル	
Linux*	icc -c -ipo main.c func1.c func2.c
Windows*	icl -c /Qipo main.c func1.c func2.c

リンク	
Linux	icc -ipo main.o func1.o func2.o
Windows	icl /Qipo main.o func1.o func2.o

XLSOFT
Copyright© 2011, XLsoft K.K. All rights reserved.

インライン展開の例
均一な二次元電荷分布により静電位を計算

呼び出し元に関数のコードを展開

```
for (i=1;i<nx;i++) {
   x = x0 + i*h;
   sumx = sumx +  func(x,y,xp,yp);
 }
```

```
float func(float x, float y, float xp, float yp)
{
  float denom;
  denom = (x-xp)*(x-xp) + (y-yp)*(y-yp);
  denom = 1./sqrt(denom);
  return denom; }
```

```
for (i=1;i<nx;i++) {
   x = x0 + i*h;
   sumx = sumx +1./sqrt((x-xp)*(x-xp) + (y-yp)*(y-yp));
 }
```

XLSOFT

Copyright© 2011, XLsoft K.K. All rights reserved.

目次

- 効率的なベクトル化、最適化のためのコード記述方法
  - ベクトル化の仕様
  - キャッシュライン
  - 配列のアラインメント
  - 構造体のアラインメント

- プロシージャ間の最適化

- プロファイルに基づく最適化 (PGO)

XLSOFT
Copyright© 2011, XLsoft K.K. All rights reserved.

プロファイルに基づく最適化
Profile Guided Optimization (PGO)

- 実行時のフィードバック情報を使用して最適化を行なう
  命令キャッシュ、ページング、分岐予測を支援
- 有効な最適化:
  - マルチパスの最適化、基本ブロックの並び替え
  - 適切なレジスターの割り当て
  - 関数のインライン化における的確な判断
  - 関数の並び替え
  - スイッチ文の最適化
  - ベクトル化における的確な判断

実行時に取得したデータ

次のコンパイルに反映して最適化

Copyright© 2011, XLsoft K.K. All rights reserved.

プロファイルに基づく最適化: -prof_gen/-prof_use（Profile-Guided Optimization: PGO）

- アプリケーションの実行プロファイル情報に基づく最適化
  - 分岐予測ミスの低減、コードサイズの縮小など
  - 効率性を考慮した自動ベクトル化や関数のインライン展開

Step 1

分析コードの埋め込み（-prof_gen）
icc -prof_gen prog.c

→ インストルメント済み実行ファイル: prog.out

Step 2

インストルメント済み実行ファイルを実行（典型的なデータセットを使用）

→ 動的プロファイル情報: 12345678.dyn ※

Step 3

フィードバックコンパイル（-prof_use）
icc -prof_use prog.c

→ 動的情報サマリーファイル: pgopti.dpi

→ 最適化済み実行ファイル: prog.out

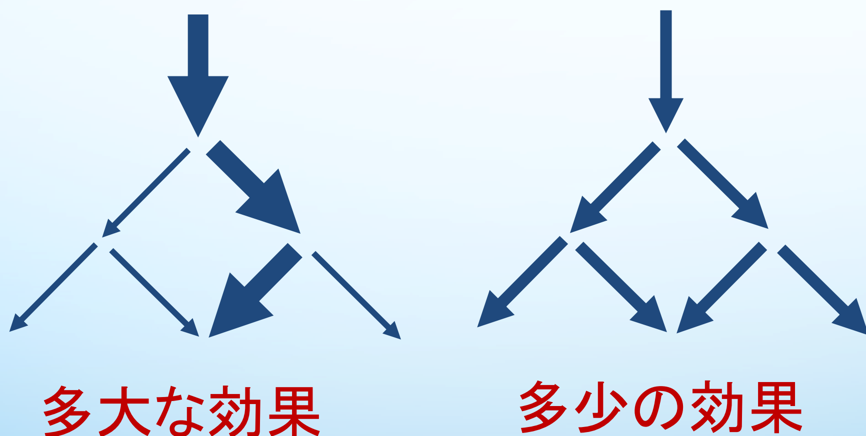
※ 複数の .dynファイルがある場合は平均化される

XLSOFT
Copyright© 2011, XLsoft K.K. All rights reserved.

PGO: 利点を得られるプログラム例
マルチパスの最適化

- 一貫性のあるホットパス
- 多数の if 文または switch文
- ネストされた if 文または switch文

CPUに読み込まれても実行されないコードはオーバーヘッドになり遅延を招く (結果的に分岐予測ミス)




```
for (i=0; i < NUM_BLOCKS; i++)
{
    switch (check3(i))   {
        case 3:          /* 25% */
            x[i] = 3;  break;
        case 10:         /* 75% */
            x[i] = 10;  break;
        default:         /* 0% */
            x[i] = 99;  break;  }
}
```

多大な効果のあるコード例

※CPUは書いてある通りに読み込む

この場合 PGOは case 3 と case 10 を入れ替え、75%の確立で正しい分岐を実行させる

Copyright© 2011, XLsoft K.K. All rights reserved.

最適化に関する注意事項

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Optimization Notice

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

Copyright© 2011, XLsoft K.K. All rights reserved.