JAXAスーパーコンピュータシステム
（JSS:Jaxa Supercomputer System）
Mシステム　チューニングガイド

１．０版

2009年7月14日

情報・計算工学センター
JAXA’s Engineering Digital Innovation Center

目次

はじめに

JAXA Supercomputer System

■ 本ガイドの対象範囲

本書は、JAXA統合スーパーコンピュータシステムの大規模並列スカラ計算機（Mシステム）上で、ユーザプログラムの性能評価およびチューニングを行う際のガイドとしてご利用頂くことを目的としています。

また、プログラミング言語は基本的にFortran及びXPFortranを対象としています。

■ 注意事項

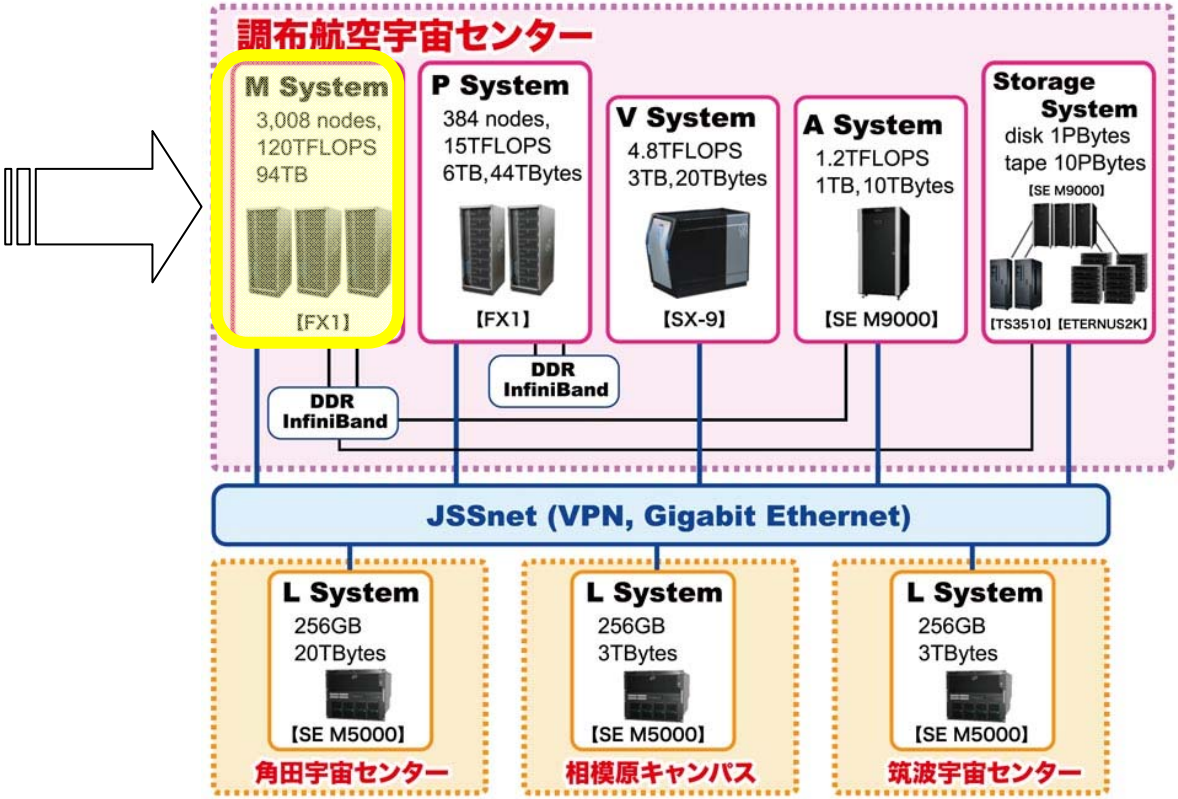
FUJITSU

本資料は，富士通株式会社の協力の下に作成されたものです。

1. 性能チューニングについて

1．1．性能チューニングとは

Mシステムは3008台の計算ノードを持つ大規模並列スカラ計算機です。ユーザプログラムを高速に実行するためには、計算ノードのハードウェア・ソフトウェアの性能を最大限に引き出す作業が必要です。このような作業を性能チューニングといいます。

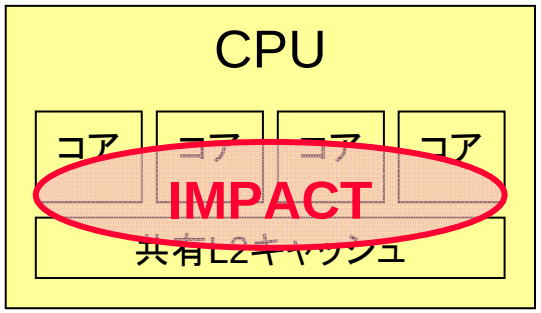


1．2．Mシステム計算ノードの特長

JAXA Supercomputer System

マルチコアCPU、高並列に対応する以下の機能を備えています。

- Integrated Multicore Parallel ArChiTecture (以降、**IMPACT**と略す)
  - ●CPU内共有L2キャッシュ
  - ●CPU内高速バリア
  - ●自動並列化コンパイラ



⇒マルチコアCPUを仮想シングルCPUとして使用

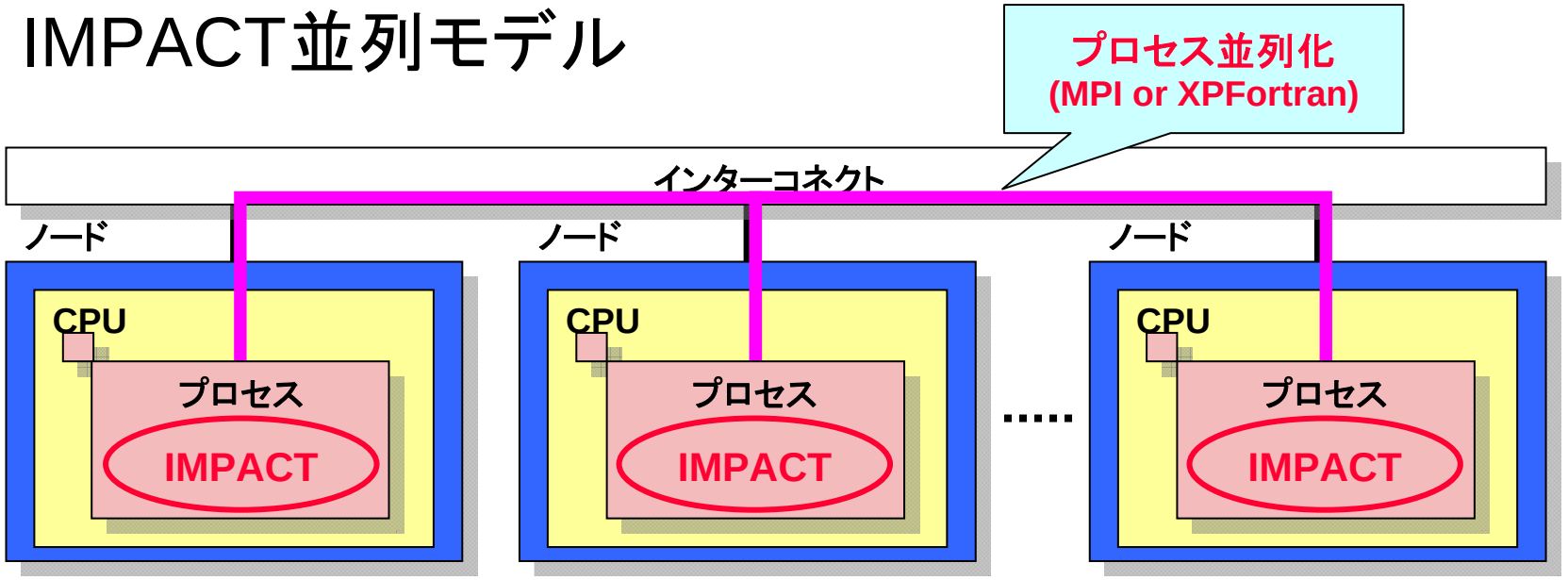
- 高機能スイッチ
  - ●MPIやXPFortranのバリア同期を高速に実行
  - ●MPIにおけるリダクション処理を高速に実行

⇒高並列実行の通信オーバーヘッドを削減

1．3．計算ノードの並列プログラミングモデル

JAXA Supercomputer System

■ IMPACT並列モデル

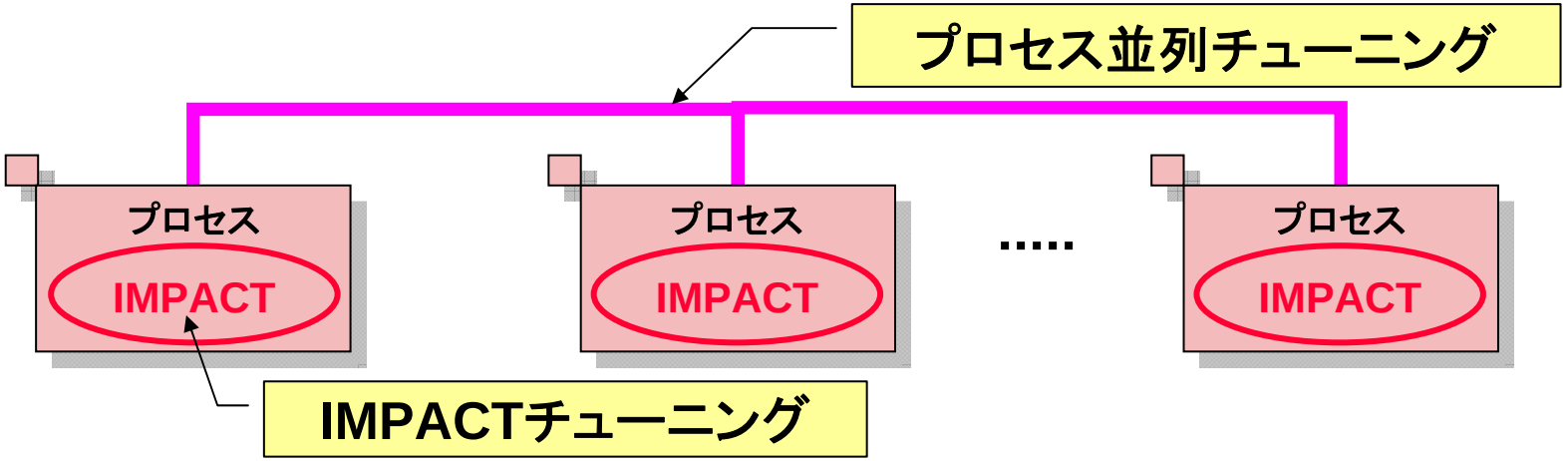


プログラムはプロセスを単位として動作します。
プロセス内部はIMPACTにより処理を自動並列化して実行します。
また、MPIやXPFortranを使用すると、複数のプロセスに分割して実行することができます（これをプロセス並列化といいます）。

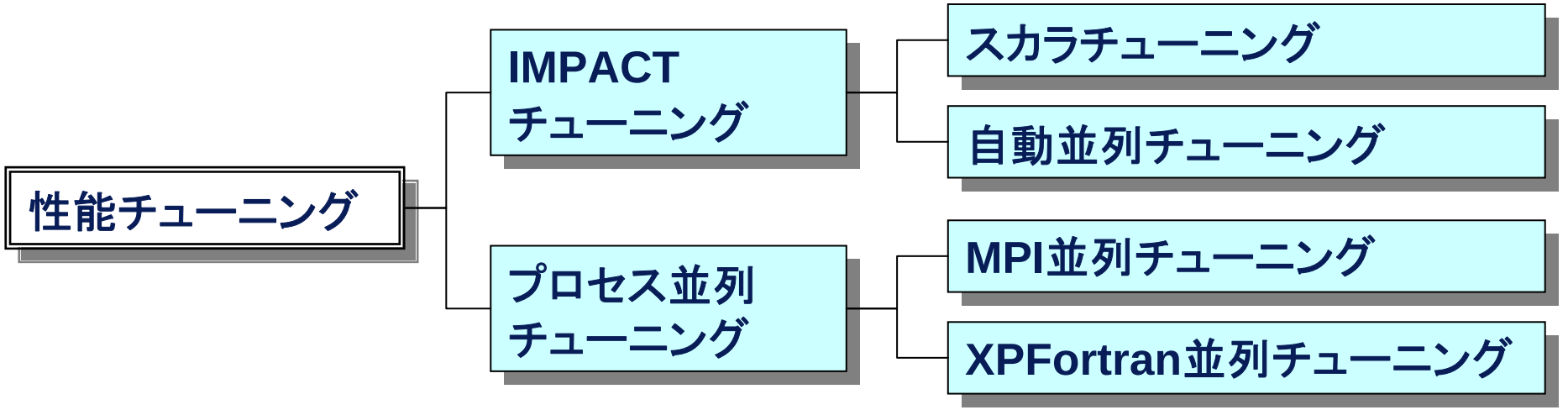
IMPACTとプロセス並列化を上図のように組合わせることで、複数ノードを使用する大規模なプログラムを実行することができます。

1．4．性能チューニングの概要

プログラミングモデルのどの部分を対象とするかによって、性能チューニングは大きく2種類に分けられます。



性能チューニング体系を以下に示します。



1．5．性能チューニングのポイント(1)

JAXA Supercomputer System

性能チューニングの種類ごとのポイントについて説明します。

- スカラチューニングのポイント
  - データの局所性を高める
  - 演算器の実行効率を高める

- 自動並列チューニングのポイント
  - 並列化率を上げる
  - 並列化粒度を上げる
  - ロードバランスを均等化させる

1．6．性能チューニングのポイント(2)

JAXA Supercomputer System

- MPI並列チューニングのポイント
  - ロードバランスを均等化させる
  - プロセス間の通信コストを削減する

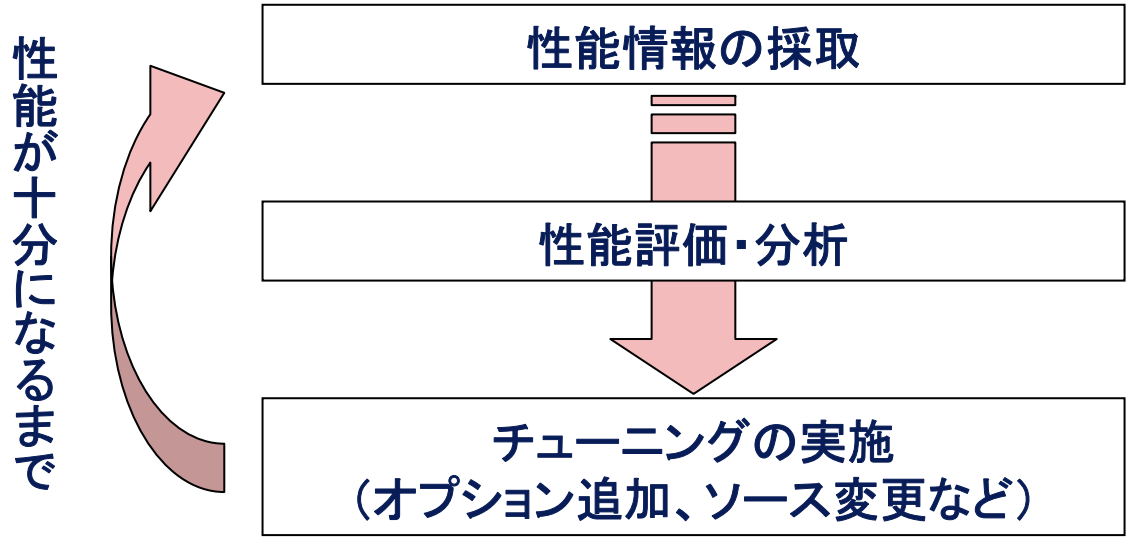
- XPFortran並列チューニングのポイント
  - 並列化率を上げる
  - 並列化粒度を上げる
  - ロードバランスを均等化させる
  - プロセス間の通信コストを削減する

2. 性能チューニングの流れ

2．1．性能チューニングの流れ

JAXA Supercomputer System

性能チューニングは以下のフェーズの繰り返しにより行います。



- **性能情報の採取**

  性能分析ツール（プロファイラ）を使用して、プロファイラ情報を採取します。

- **性能評価・分析**

  性能の目安値と性能情報を比較し、不十分ならば、ソースやプロファイラ情報から原因を分析します。

- **チューニングの実施**

  翻訳/実行オプションを追加したり、ソースコードを変更することにより、性能を改善します。

2．2．性能分析の目的

JAXA Supercomputer System

性能分析の目的は、プログラム中のホットスポット（多くの実行時間を要する箇所）を探すことにあります。

■ アムダールの法則（Amdahl's Law）

プログラム全体の実行時間のp（割合：0≦p≦1）にあたる部分をn倍向上させた場合の、プログラム全体の性能向上比は以下の通り。

$$性能向上比 = \frac{1}{(1-p)+\frac{p}{n}}$$

- プログラム全体の90%にあたる処理を10倍性能向上 ⇒ 性能向上比＝約5.26倍
- プログラム全体の30%にあたる処理を10倍性能向上 ⇒ 性能向上比＝約1.37倍

したがって、性能チューニングでは、採取したプログラムの性能情報から、ホットスポットの特定および分析を行うことが重要になります。

３．性能情報採取

3．1．プロファイラの概要

JAXA Supercomputer System

プロファイラは、プログラムの性能分析を定量的に行うために有用な情報を収集するツールです。

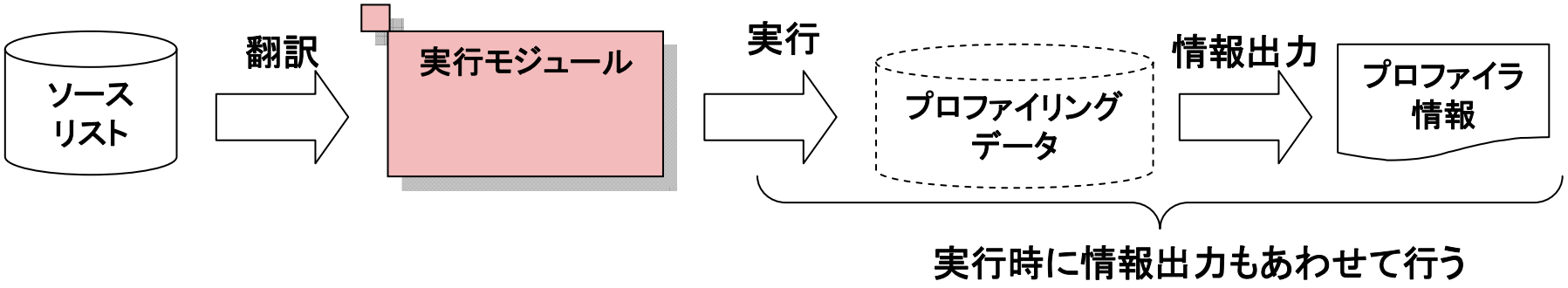
機能概要

プロファイラは以下のような性能情報を出力することができます。

- 経過時間およびCPU時間情報
- プロセス間通信の時間情報
- サンプリングによるコスト分布情報
- 実行時のハードウェア動作状況

動作概要

ユーザが以下の手順を段階的に行うことにより使用します。



3．2．プロファイラによる情報採取手順

以下にプロファイラ情報の採取手順を示します。

①翻訳

収集対象プログラムをコンパイラで翻訳します。

```
f90jx    sample.f90  -o  sample.exe
```

②実行とプロファイラ情報の出力

出力ファイル名と実行モジュールを指定してプロファイラコマンド（**fpcoll**）を実行すると、プログラム実行と同時に情報収集を行い、結果をファイルに出力します。

```
fpcoll -Icall,balance,hwm -l30 -o result.txt ./sample.exe
```

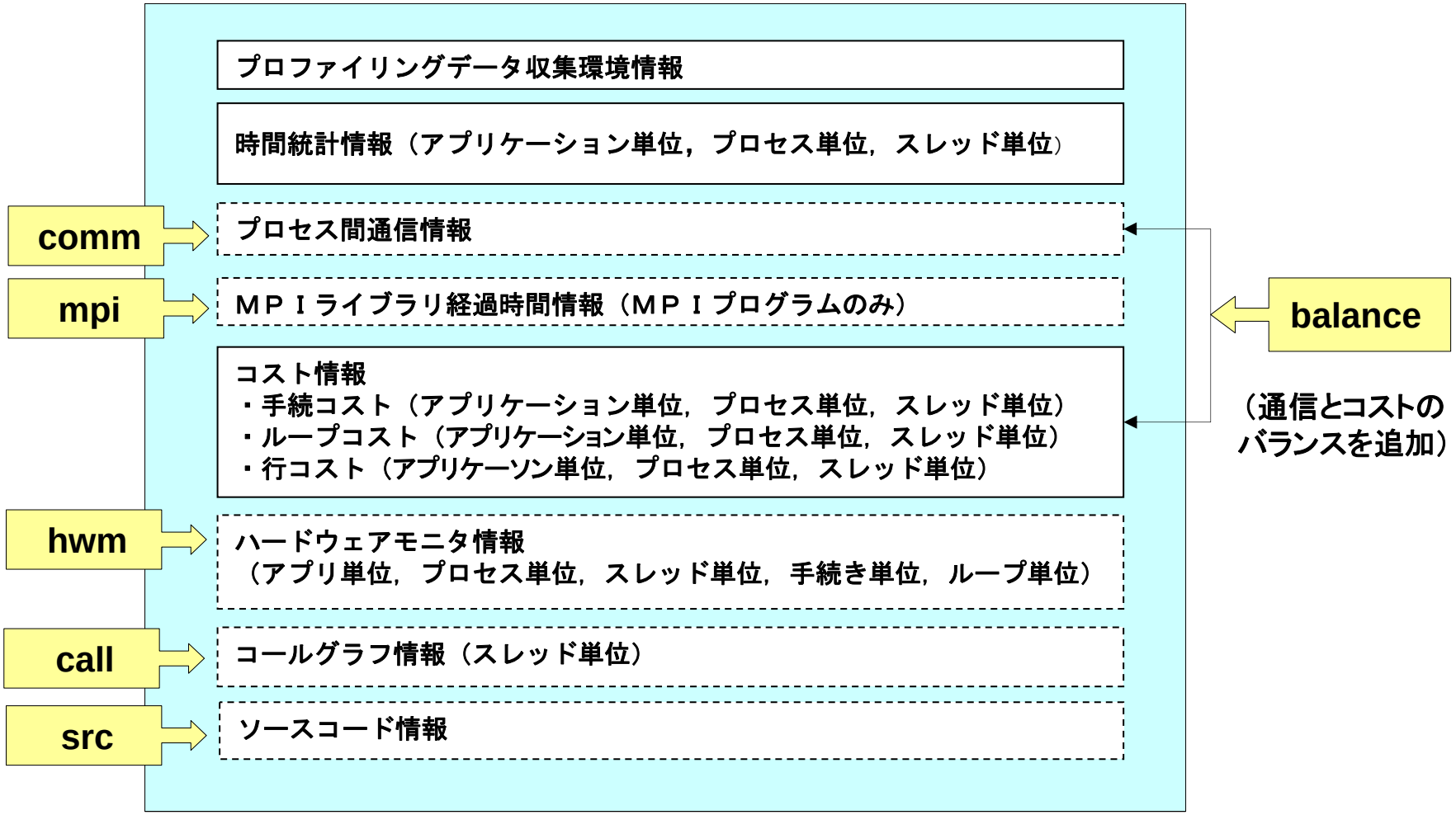
- 収集および出力項目
- 手続情報の出力件数
- プロファイラ情報出力ファイル名
- ①で生成した実行モジュール

プロセス並列プログラムの場合は、翻訳・実行時に並列用オプションを指定して、同様の方法で情報を取得します。

3．3．プロファイラ採取情報の見方(1)

JAXA Supercomputer System

fpcollコマンドで出力されるプロファイラ情報の構成は以下の通りです。



3．4．プロファイラ情報の見方（2）

JAXA Supercomputer System

■ 性能情報について

プロファイラが出力する情報のうち、性能チューニングの指標となる代表的な性能情報は以下の通りです。

性能情報	内容	性能目安
MIPS値	命令の実行効率 （命令実行数÷CPU時間÷1e+6）	4000以上 （1プロセスあたり）
MFLOPS値	浮動小数点命令実行速度 （浮動小数点演算命令数÷CPU時間÷1e+6）	2000以上 （1プロセスあたり）
L2キャッシュミス率	2次キャッシュのミスヒット率 （2次キャッシュミス回数÷命令実行数）[%]	0．2%未満 （1プロセスあたり）

これらの情報はハードウェアモニタ情報から参照できます。

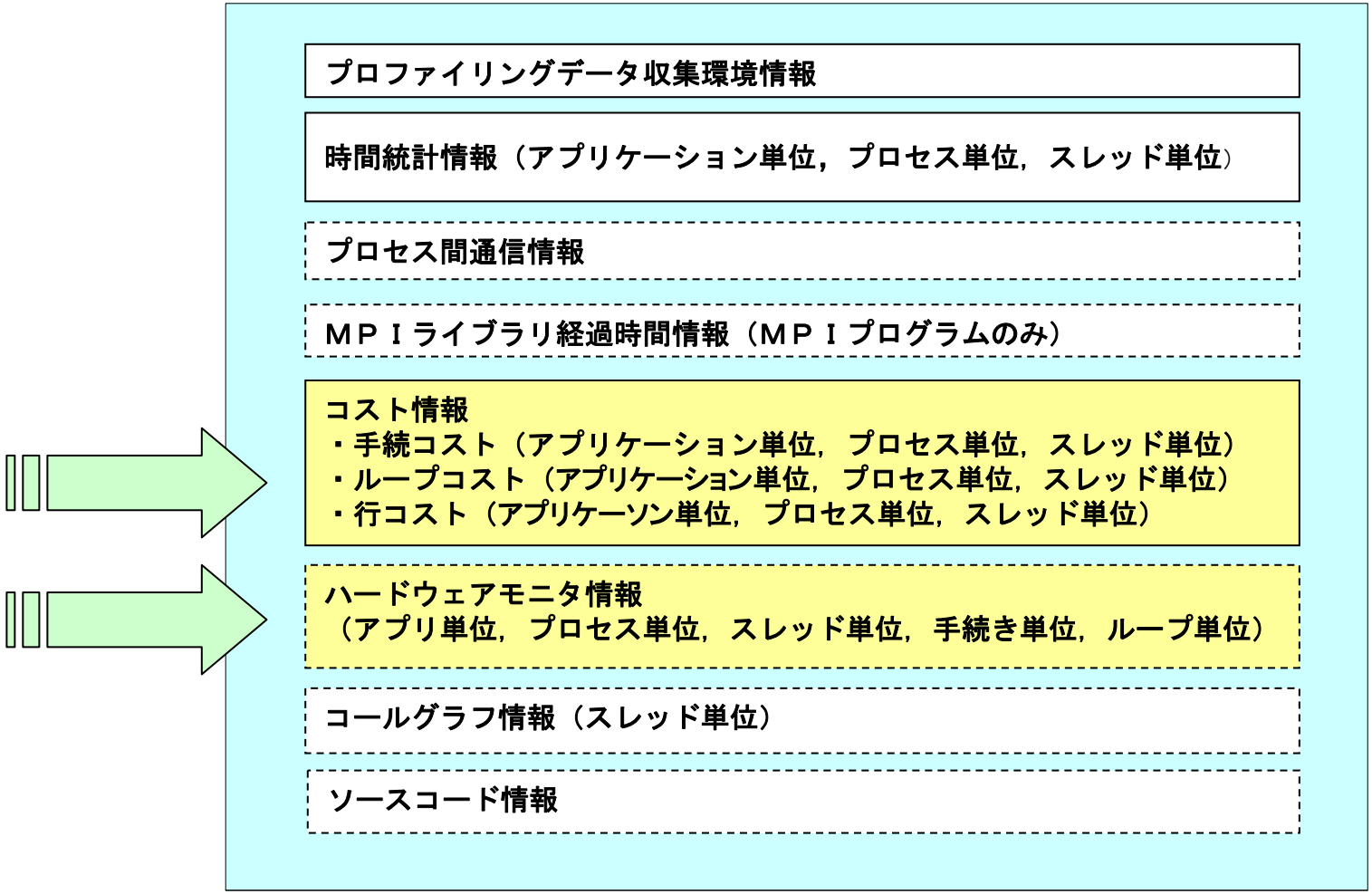
出力値が性能目安に満たない場合はチューニングが必要です。

4. スカラチューニング

4．1．スカラ性能の評価（1）

JAXA Supercomputer System

スカラチューニングのためのホットスポット特定および分析を行うには、
プロファイラ情報の以下の箇所に注目します。



4. 2. スカラ性能の評価(2)

■ コスト情報･･･ホットスポットの抽出

コスト

コスト比率

```
********************************************************************************
    Process        1 - procedures
********************************************************************************
```

Cost	%		%	Start	End	
16873	100.0000	210	1.2446	--	--	Process 1
1563	9.2633	0	0.0000	57	72	sub1._PRL_1_
1463	8.6707	0	0.0000	183	191	sub2._PRL_1_
1462	8.6647	0	0.0000	231	237	sub3._PRL_3_
1381	8.1847	0	0.0000	885	890	sub4._PRL_4_
1293	7.6631	0	0.0000	223	229	sub5._PRL_1_
1074	6.3652	0	0.0000	259	388	sub6

手続き名＋行番号のリスト
後半の「_PRL_」はIMPACTで自動並列化されたことを、「_OMP_」はOpenMPで並列化されたことを示す

コスト比率が高いところは、それだけ全体時間に対して多くの実行時間を要する箇所＝ホットスポットになります。

コスト情報の上位に現れる手続き名とその行番号に注目します。

4．3．スカラ性能の評価(3)

JAXA Supercomputer System

ハードウェアモニタ情報・・・性能指標の判定

MIPS値

MFLOPS値

```
**********************************************************************
    Process      1 - performance monitors
**********************************************************************

                 Time(s)   Instructions          MIPS        MFLOPS     Cover(%)   Start      End
----------------------------------------------------------------------------------------------------
   Elapsed       83.5100   292547367851     3503.1428      770.6741      98.4849      --       --   Process       1
----------------------------------------------------------------------------------------------------
   Elapsed        4.3556    26445838686     6071.6699     1470.7395      98.2867      57       72   sub1._PRL_1_
   Elapsed        5.2387    23881767634     4558.6968      824.4698      98.3642     183      191   sub2._PRL_1_
   Elapsed        4.0266    20498623749     5090.8169     1259.5062      98.3919     885      890   sub3._PRL_4_
   Elapsed        3.6865    18813359810     5103.2896     1332.8138      98.7677     231      237   sub4._PRL_3_
   Elapsed        3.6244    18026236014     4973.6074     1264.7723      98.6891     223      229   sub5._PRL_1_
   CPU           11.2920    34802192952     3082.0183      580.0624      98.9115     259      388   sub6
```

L2キャッシュミス率

```
                 Time(s)     L2 miss(%)     mTLB-is(%)     mTLB-op(%)     Cover(%)   Start      End
----------------------------------------------------------------------------------------------------
   Elapsed       83.5100         0.3323         0.0000         0.0000      98.4849      --       --   Process       1
----------------------------------------------------------------------------------------------------
   Elapsed        4.3556         0.3377         0.0000         0.0000      98.2867      57       72   sub1._PRL_1_
   Elapsed        5.2387         0.4233         0.0000         0.0000      98.3642     183      191   sub2._PRL_1_
   Elapsed        4.0266         0.2947         0.0000         0.0000      98.3919     885      890   sub3._PRL_4_
   Elapsed        3.6865         0.2825         0.0000         0.0000      98.7677     231      237   sub4._PRL_3_
   Elapsed        3.6244         0.2875         0.0000         0.0000      98.6891     223      229   sub5._PRL_1_
   CPU           11.2920         0.3243         0.0000         0.0000      98.9115     259      388   sub6
```

性能目安に満たないものがチューニング対象になります。

４．４．スカラ性能の分析

採取したプロファイラ情報が以下の判定条件に該当する場合は、
対応するポイントの性能チューニングを行います。

プロファイラ結果の判定条件	スカラチューニングのポイント
L2キャッシュミス率が0．2%を超えるルーチンがある	・キャッシュミス率を下げる
MFLOPS値が2000未満のルーチンがある	・演算器の実行効率を高める

4．5．スカラチューニング手法

JAXA Supercomputer System

■ オプションチューニング

コンパイラには、標準オプションよりも強力な最適化を指示するオプションがいくつか提供されています。その中でもスカラ性能を向上させる可能性のある代表的なオプションについて、以下にご紹介します。

チューニング内容	対象となる翻訳オプション	最適化の効果
配列のパディング	-Karraypad_const -Karraypad_expr	キャッシュ競合の削減
配列の次元入替え	-Karray_subscriptなど	キャッシュ利用効率の向上
配列の融合	-Karray_mergeなど	キャッシュ利用効率の向上
インダイレクトアクセスのプリフェッチ	-Kprefetch_indirect	メモリアクセスの高速化
IF構文に含まれるアクセスのプリフェッチ	-Kprefetch_conditional	メモリアクセスの高速化

効果の有無は、プログラム・データ特性によって変わりますので、実際にオプションを付けて翻訳した実行モジュールで確認する必要があります。

4．6．オプションチューニング事例（1）

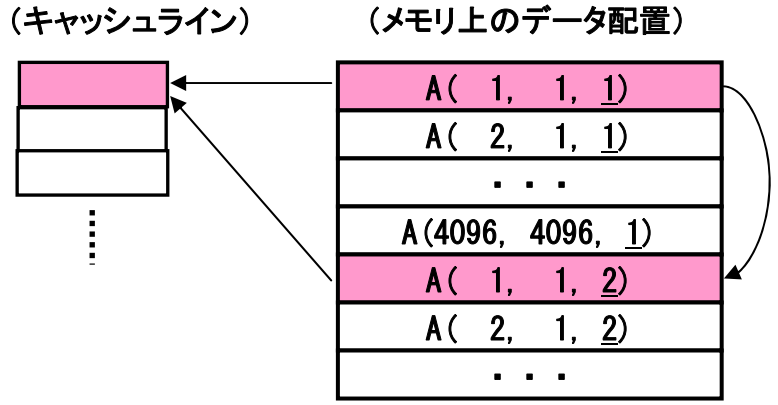
JAXA Supercomputer System

■ コンパイラによる配列のパディング

配列の形状が2のべき乗になっていると、キャッシュ競合で性能低下する場合があります。
パディング（内側にすき間を空ける）を行うことにより、キャッシュ競合を削減します。

オリジナルソース

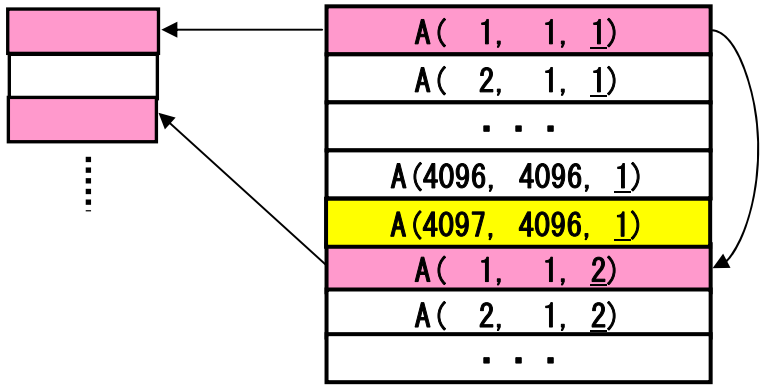
```
REAL*8 A(4096, 4096, 3)
DO I=1,N
A(I,1,1)=A(I,1,2)+A(I,1,3)
ENDDO
```



同じキャッシュラインへの競合

コンパイラ最適化のイメージ

```
REAL*8 A(4097, 4096, 3)
DO I=1,N
A(I,1,1)=A(I,1,2)+A(I,1,3)
ENDDO
```



異なるキャッシュラインで競合回避

4．7．オプションチューニング事例（2）

JAXA Supercomputer System

■ 配列パディングの翻訳オプション

翻訳時オプション形式：　-Karraypad_const[=N]
　　-Karraypad_expr=N

◆対象配列を自動選出→パディングを適用

◆選出基準は拡張オプションで選択（どちらか一方のみ）

翻訳時オプション	対象範囲
-Karraypad_const[=N]	1次元目が明示上下限で定数式の配列にN要素のパディングを適用　(1≦N≦2,147,483,647) N省略時はコンパイラが適切な値を設定
-Karraypad_expr=N	1次元目が明示上下限の配列にN要素のパディングを適用　(1≦N≦2,147,483,647)

◆注意事項

- ●対象配列を使うソース全てにオプション指定が必要です。
- ●パディングの効果はプログラムによって異なります。
- ●配列結合を使うプログラムなどでは計算結果が異なる場合があります。
- ●XPFortran(-Uxpf)と併用できません。

4．8．オプションチューニング事例（3）

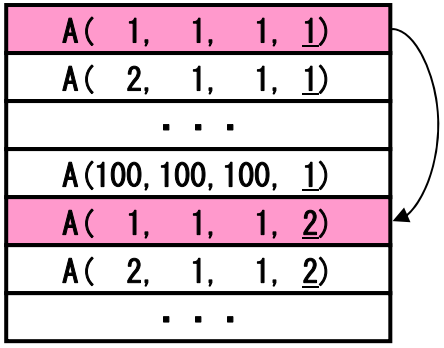
JAXA Supercomputer System

コンパイラによる配列次元移動

配列次元の外側が変化するアクセスの場合、キャッシュの利用効率が下がって性能低下する場合があります。変化する次元を内側に移動することにより、データアクセスを連続化して、キャッシュ利用効率を向上させます。

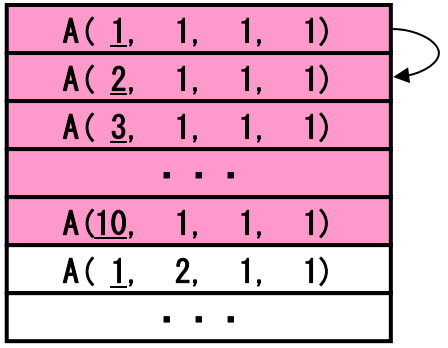
オリジナルソース

```
   REAL*8 A(100,100,100,10)
   DO 10 K=1,100
   DO 10 J=1,100
   DO 10 I=1,100
     A(I,J,K,1)=A(I,J,K,2)+A(I,J,K,3)+
              ･･･+A(I,J,K,10)
10 CONTINUE
```



コンパイラ最適化のイメージ

```
   REAL*8 A(10,100,100,100)
   DO 10 K=1,100
   DO 10 J=1,100
   DO 10 I=1,100
     A(1,I,J,K)=A(2,I,J,K)+A(3,I,J,K)+
              ･･･+A(10,I,J,K)
10 CONTINUE
```



4．9．オプションチューニング事例（4）

JAXA Supercomputer System

■ 配列次元移動の翻訳オプション

翻訳時オプション形式：
-Karray_subscript
-Karray_subscript_elementlast=N
-Karray_subscript_element=N
-Karray_subscript_rank=N
-Karray_subscript_variable='ary_nm(rank,rank[,rank…])'

◆対象配列を自動選出→最外次元を最内に移動

◆選出基準は拡張オプションで調整

選出基準	-Karray_subscriptのみ	-Karray_subscript + 拡張オプション
最外次元要素数（上限値）	10以下	-Karray_subscript_elementlast=N (2≦N≦2,147,483,647)
最外以外の次元要素数（下限値）	100以上	-Karray_subscript_element=N (2≦N≦2,147,483,647)
配列の次元数（下限値）	4以上	-Karray_subscript_rank=N (2≦N≦30)
配列変数名を直接指定	（コンパイラが自動選出）	-Karray_subscript_variable = 'ary_nm(rank,rank[rank…]) (1≦rank≦30）

4．10．オプションチューニング事例(5)

JAXA Supercomputer System

使用例）配列形状が(150,100,100,11)の場合

```
real*8 a(150,100,100,11),b(150,100,100,11),c(150,100,100,11)
do k=1,100
do j=1,100
do i=1,150
  a(i,j,k,1)=a(i,j,k,2)+a(i,j,k,3)+a(i,j,k,4)+a(i,j,k,5)+・・・+a(i,j,k,10)+a(i,j,k,11)
  b(i,j,k,1)=b(i,j,k,2)+b(i,j,k,3)+b(i,j,k,4)+b(i,j,k,5)+・・・+b(i,j,k,10)+b(i,j,k,11)
  c(i,j,k,1)=c(i,j,k,2)+c(i,j,k,3)+c(i,j,k,4)+c(i,j,k,5)+・・・+c(i,j,k,10)+c(i,j,k,11)
end do
end do
end do
```

```
指定オプション： -Karray_subscript,array_subscript_elementlast=11
翻訳メッセージ：
 jwd2490i-i  "test.f", line 1: 配列変数'c'の添字を指定された順序に変更しました.
 jwd2490i-i  "test.f", line 1: 配列変数'b'の添字を指定された順序に変更しました.
 jwd2490i-i  "test.f", line 1: 配列変数'a'の添字を指定された順序に変更しました.
```

◆注意事項

- ●対象配列を使うソース全てにオプション指定が必要です。
- ●移動の効果はプログラムによって異なります。また、計算結果が異なる場合もあります。
- ●デバッグオプション(-gおよび-Haesux)と併用できません。
- ●XPFortran(-Uxpf)と併用できません。

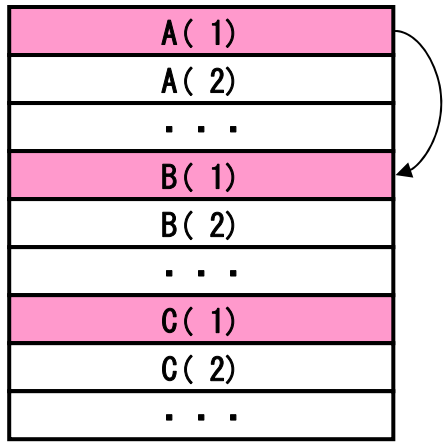
4．11．オプションチューニング事例(6)

JAXA Supercomputer System

コンパイラによる配列融合

同一ループ内でアクセスパターンが共通の配列が複数ある場合、1個の配列に融合することにより、データアクセスを連続化して、キャッシュ利用効率を向上させます。

オリジナルソース	コンパイラ最適化のイメージ
REAL*8 A(N),B(N),C(N) DO 10 I=1,N A(I)=B(I)+C(I) 10 CONTINUE	REAL*8 ABC(**3**,N) DO 10 I=1,100 ABC(**1**,I)=ABC(**2**,I)+ABC(**3**,I) 10 CONTINUE



4．12．オプションチューニング事例(7)

JAXA Supercomputer System

■ 配列融合の翻訳オプション

翻訳時オプション形式： -Karray_merge
-Karray_merge_common[=name]
-Karray_merge_local

◆対象範囲内の複数配列をマージ

◆対象範囲はオプションで選択

翻訳時オプション	対象範囲
-Karray_merge	全コモンブロック+ローカル変数
-Karray_merge_common[=name]	コモンブロック内(nameはコモンブロック名) name省略時は全コモンブロックが対象
-Karray_merge_local	ローカル変数

◆注意事項

- ●対象配列を使うソース全てにオプション指定が必要です。
- ●融合の効果はプログラムによって異なります。また、計算結果が異なる場合もあります。
- ●デバッグオプション(-gおよび-Haesux)と併用できません。
- ●XPFortran(-Uxpf)と併用できません。

4．13．オプションチューニング事例(8)

JAXA Supercomputer System

■ コンパイラによるインダイレクトアクセスのプリフェッチ

標準のプリフェッチ対象に含まれない、配列のリスト参照のようなインダイレクトアクセスについてもプリフェッチ命令を生成することで、メモリアクセスを高速化します。

オリジナルソース	コンパイラ最適化のイメージ
REAL*8 A(N),B(N),C(N) REAL*8 D(N),E(N),F(N) DO 10 I=1,N **配列D,E,Fに対して プリフェッチ命令を生成** A(D(I))=B(E(I))+SCALAR*C(F(I)) 10 CONTINUE	REAL*8 A(N),B(N),C(N) REAL*8 D(N),E(N),F(N) DO 10 I=1,N **配列A,B,C,D,E,Fに対して プリフェッチ命令を生成** A(D(I))=B(E(I))+SCALAR*C(F(I)) 10 CONTINUE

4．14．オプションチューニング事例（9）

JAXA Supercomputer System

■ インダイレクトアクセスに関するプリフェッチの翻訳オプション

翻訳時オプション形式： **-Kprefetch_indirect**

◆間接アクセスされる配列データに対してプリフェッチ命令を生成

◆注意事項

- ●プリフェッチの効果はプログラムによって異なります。
- ●プリフェッチのために命令数が増加するため、性能低下する可能性もあります。

4．15．オプションチューニング事例（10）

JAXA Supercomputer System

■ コンパイラによるIF構文内部アクセスのプリフェッチ

標準のプリフェッチ対象に含まれない、IF構文やCASE構文内部のアクセスについても、　プリフェッチ命令を生成することで、メモリアクセスを高速化します。

オリジナルソース	コンパイラ最適化のイメージ

REAL*8 A(N),B(N),C(N)
REAL*8 D(N),E(N),F(N)
DO 10 I=1,N
**配列D,E,Fに対して
プリフェッチ命令を生成**
D(I)=E(I)+SCALAR*F(I)
IF(I<M)THEN
A(I)=B(I)+SCALAR*C(I)
ENDIF
10 CONTINUE
	REAL*8 A(N),B(N),C(N)
	REAL*8 D(N),E(N),F(N)
	DO 10 I=1,N
	**配列A,B,C,D,E,Fに対して
	プリフェッチ命令を生成**
	D(I)=E(I)+SCALAR*F(I)
	IF(I<M)THEN
	A(I)=B(I)+SCALAR*C(I)
	ENDIF
	10 CONTINUE

4．16．オプションチューニング事例（11）

JAXA Supercomputer System

■ IF構文に含まれるアクセスに関するプリフェッチの翻訳オプション

翻訳時オプション形式： -Kprefetch_conditional

◆IF構文やCASE構文に含まれるブロックの中で使用される配列データに対してプリフェッチ命令を生成

◆注意事項

●プリフェッチの効果はプログラムによって異なります。

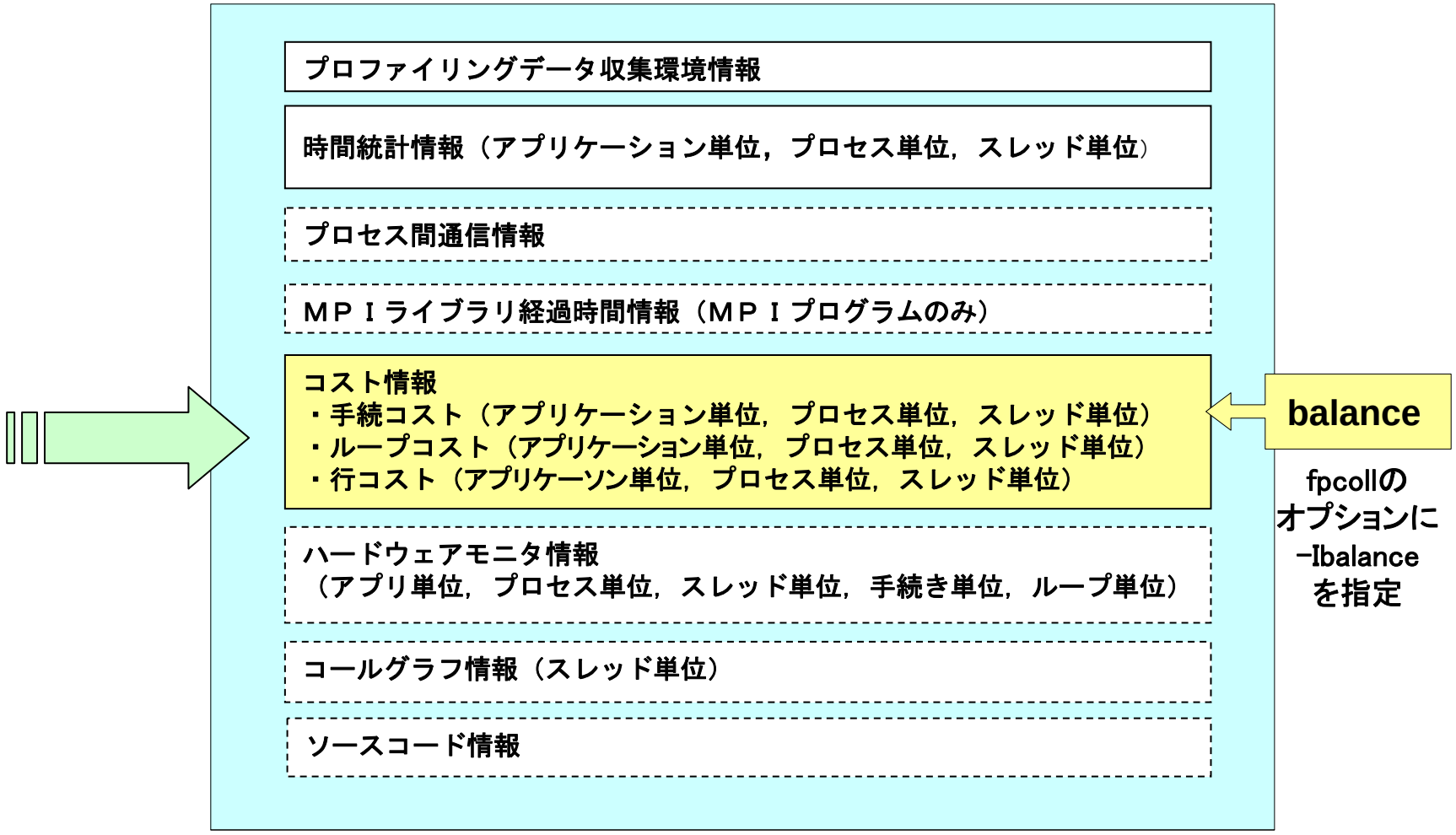
●IF文の条件によってプリフェッチしたデータが使用されない場合があるため、性能低下する可能性もあります。

5. 自動並列チューニング

5．1．自動並列性能の評価(1)

JAXA Supercomputer System

自動並列性能のホットスポット特定および分析を行うには、プロファイラ情報の以下の箇所に注目します。



5．2．自動並列性能の評価（2）

■ コスト情報･･･自動並列化されていない手続きの抽出

コスト

```
********************************************************************************
    Process        1 - procedures
********************************************************************************
```

Cost	%		%	Start	End	
16873	100.0000	210	1.2446	--	--	Process
1563	9.2633	0	0.0000	57	72	sub1._PRL_1_
1463	8.6707	0	0.0000	183	191	sub2._PRL_1_
1462	8.6647	0	0.0000	231	237	sub3._PRL_3_
1381	8.1847	0	0.0000	885	890	sub4._PRL_4_
1293	7.6631	0	0.0000	223	229	sub5._PRL_1_
1074	6.3652	0	0.0000	259	388	sub6

コスト比率

手続き名＋行番号のリスト
後半の「_PRL_」はIMPACTで自動並列化されたことを、「_OMP_」はOpenMPで並列化されたことを示す

　コスト比率が高いところは、それだけ全体時間に対して多くの実行時間を要する箇所＝ホットスポットになります。

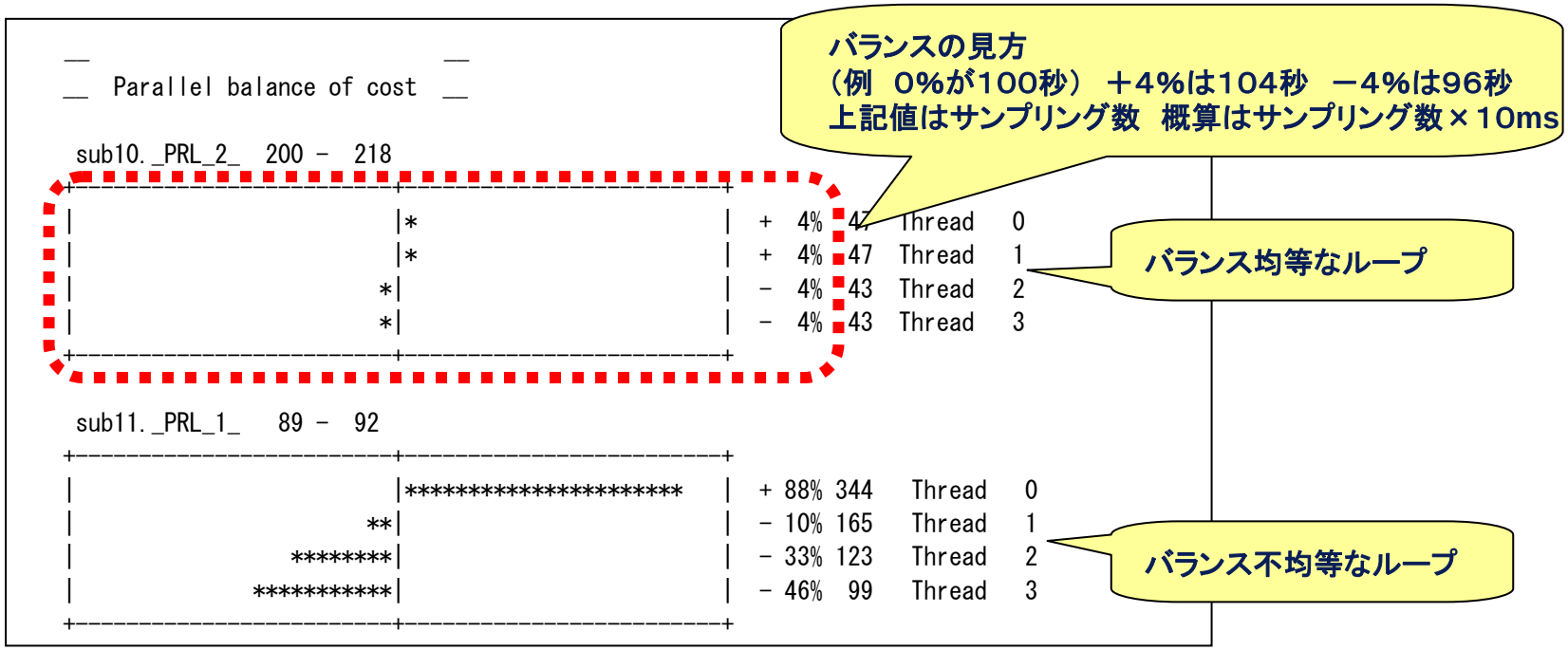
　手続き名に「_PRL_」が付いていないものは自動並列化が阻害されている可能性があります。そのため、コスト情報の上位に現れる自動並列化されていない手続き名とその行番号に注目します。

5．3．自動並列性能の評価(3)

JAXA Supercomputer System

■ バランス情報・・・ロードバランス不均等な手続きの抽出

バランス



自動並列化された手続きでも、ロードバランスが不均一なために十分な並列効果が出ない場合はホットスポットになります。

バランス情報にスレッド単位のコストが表示されますので、コスト情報の上位に現れるバランス不均等な手続き名とその行番号に注目します。

5．4．自動並列性能の分析

JAXA Supercomputer System

採取したプロファイラ情報が以下の判定条件に該当する場合は、
対応するポイントの性能チューニングを行います。

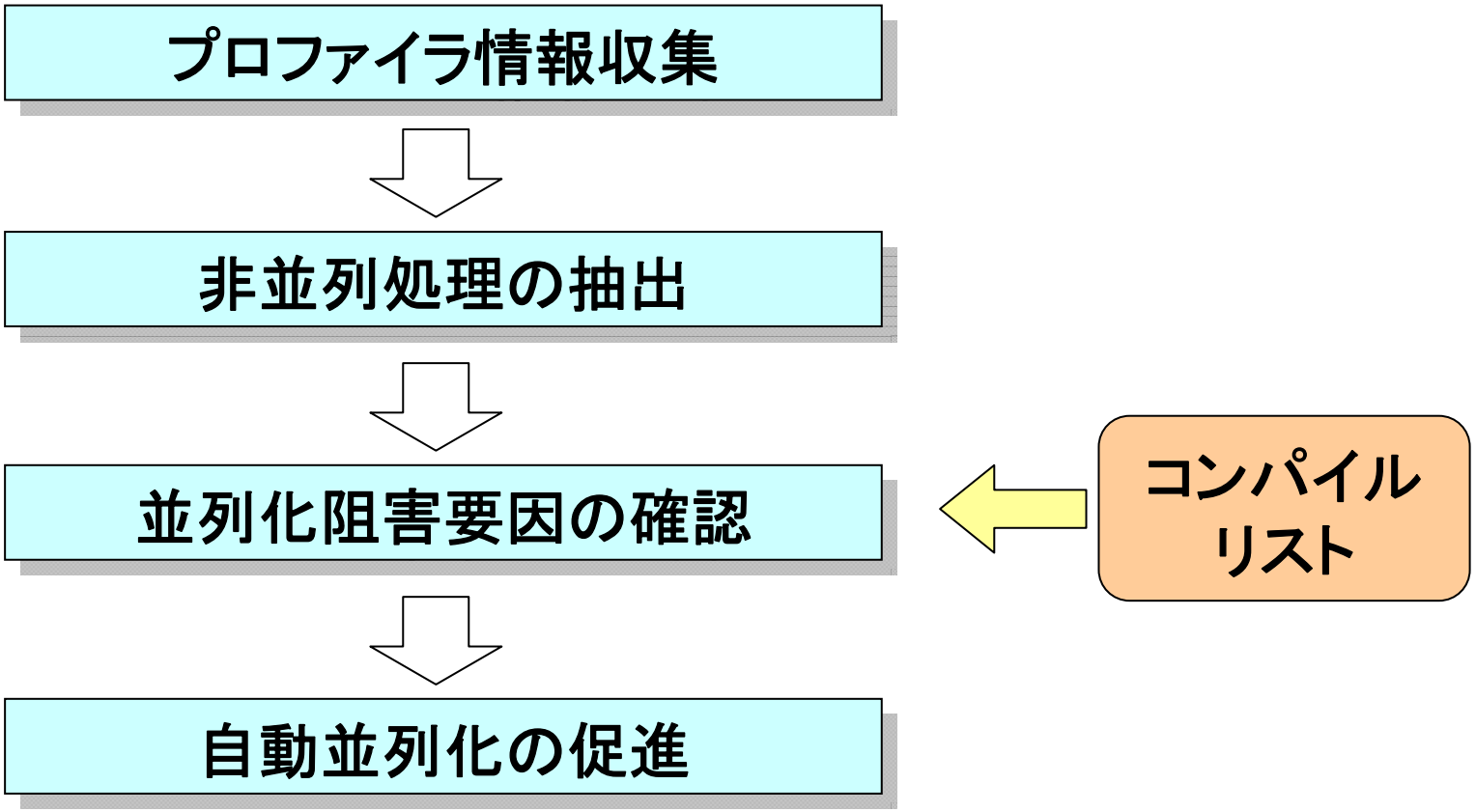
プロファイラ結果の判定条件	自動並列チューニングのポイント
自動並列化されていないルーチンがある	・並列化率を上げる ・並列化粒度を上げる
自動並列化されているが、バランスが不均等なルーチンがある	・ロードバランスを均等化させる

5．5．自動並列チューニング手法

JAXA Supercomputer System

■ コンパイルリストを利用した自動並列化の促進

翻訳時のオプション指定により、自動並列化の詳細情報を含んだコンパイルリストを採取することができます。プロファイラ情報とコンパイルリストを併用することにより、自動並列化の促進を効率的に行うことができます。



5．6．コンパイルリストの採取方法（1）

JAXA Supercomputer System

■ 診断メッセージの出力方法

翻訳時に自動並列化の状況を診断メッセージとして出力

翻訳時オプション形式：　-Kpmsg[= { 1 | 2 | 3 }]　　(Fortran/C共通)

オプション	機能
-Kpmsg=1	自動並列化した旨のメッセージのみ出力
-Kpmsg=2	-Kpmsg=1に加え、自動並列化できなかったことを示す簡略化メッセージを出力
-Kpmsg=3	-Kpmsg=1に加え、自動並列化できなかったことを示すメッセージを出力

翻訳例

```
$ f90jx –Kpmsg sample.f90
```
（レベル省略時は-Kpmsg=3）

診断メッセージ出力例

```
Diagnostic messages: program name(sample)

jwd5143i-i  “sample.f”, line 406: DOループの繰返し数が少ないため,
このDOループは並列化されません.

jwd5001i-i  "sample.f", line 695: このDOループは,並列化されました.
(名前:j)
```

5．7．コンパイルリストの採取方法（2）

JAXA Supercomputer System

■ コンパイルリストの採取方法（Fortranのみ）

翻訳時に最適化情報を含むコンパイルリストを出力

翻訳時オプション形式： -Qt (Fortran)

翻訳例

```
$ f90jx -Qt sample.f90
```

→sample.lstをコンパイルリストとして出力

```
$ f90jx –Kpmsg -Qt sample.f90
```

→sample.lstをコンパイルリストとして自動並列化状況と合わせて出力

コンパイルリスト出力情報の詳細は下記オンラインマニュアルをご参照ください。

「Fortran使用手引書」（4．1 翻訳時の出力情報）

https://www.jss.jaxa.jp/

5．8．OCL行による自動並列化促進

自動並列化促進のためにコンパイラに指示を与える方法として、
最適化制御行（OCL）が用意されています。

最適化制御行（OCL）：Optimization Control Line

```
!OCL INDEPENDENT(SUB)
      DO I=1,N
         CALL SUB(A(I))
      ENDDO
```

← 第１～５桁が“!OCL”
続けて最適化指示子を記述

コメント行の形式ですが、f90jxコマンドで翻訳すると
OCL行として有効化されます。

コンパイルリストの出力情報からOCL行を挿入する事例を
いくつかご紹介します。

5．9．OCL行の挿入事例（1）

JAXA Supercomputer System

OCL使用例1：NORECURRENCE

以下のような診断メッセージが出力される場合に使用します。

jwd5101i-i “prog.f90”, line 32: DOループ内に,自動並列化の制約となる文が存在します.

配列要素の定義・参照（依存関係）がループの回転をまたがらないことを明示します。

```
!OCL NORECURRENCE(a)
    DO i=1,n
      a(x(i)) = ・・・
      ・・・ = a(x(i))
    ENDDO
```

NORECURRENCEの指定により、依存関係が解決されループが並列化される。

x(1:N)の中に同一の値がないことを保証できる場合に指定可能

5．10．OCL行の挿入事例(2)

JAXA Supercomputer System

OCL使用例2：INDEPENDENT

以下のような診断メッセージが出力される場合に使用します。

> jwd5122i-i “prog.f90”, line 28: DOループ内に,自動並列化の制約となる手続き引用が存在します.

外部手続きの引用について、配列の定義・参照(依存関係)がループの回転をまたがらないことを明示します。

```
!OCL INDEPENDENT(SUB)
    DO I=1,N
      CALL SUB(A(I))
    ENDDO
```

← INDEPENDENTの指定により、依存関係が解決されループが並列化される。

← SUB内に依存関係がないことを保証できる場合に指定可能

5．11．翻訳オプション指定事例(1)

JAXA Supercomputer System

IMPACTの基本的な自動並列化では、翻訳時に並列化の軸を決定し、実行時にはその決定ループのみで並列実行します。

しかし、回転数が動的に決まるDOループで並列化軸の回転数が1になった場合、内側に別のループがあっても逐次実行になります。

ソース翻訳時	プログラム実行時
pp do k=1,kmax ← kmaxの値は未定 p do j=1,jmax p do i=1,imax ・・・ p end do p end do p end do **Kの軸で自動並列化**	pp do k=1,kmax ← kmax=1 p do j=1,jmax p do i=1,imax ・・・ p end do p end do p end do **並列効果が無いため逐次実行**

5．12．翻訳オプション指定事例(2)

JAXA Supercomputer System

- 多重ループの並列化軸を動的に選択する翻訳オプション

翻訳時オプション形式： **-Kdynamic_iteration**

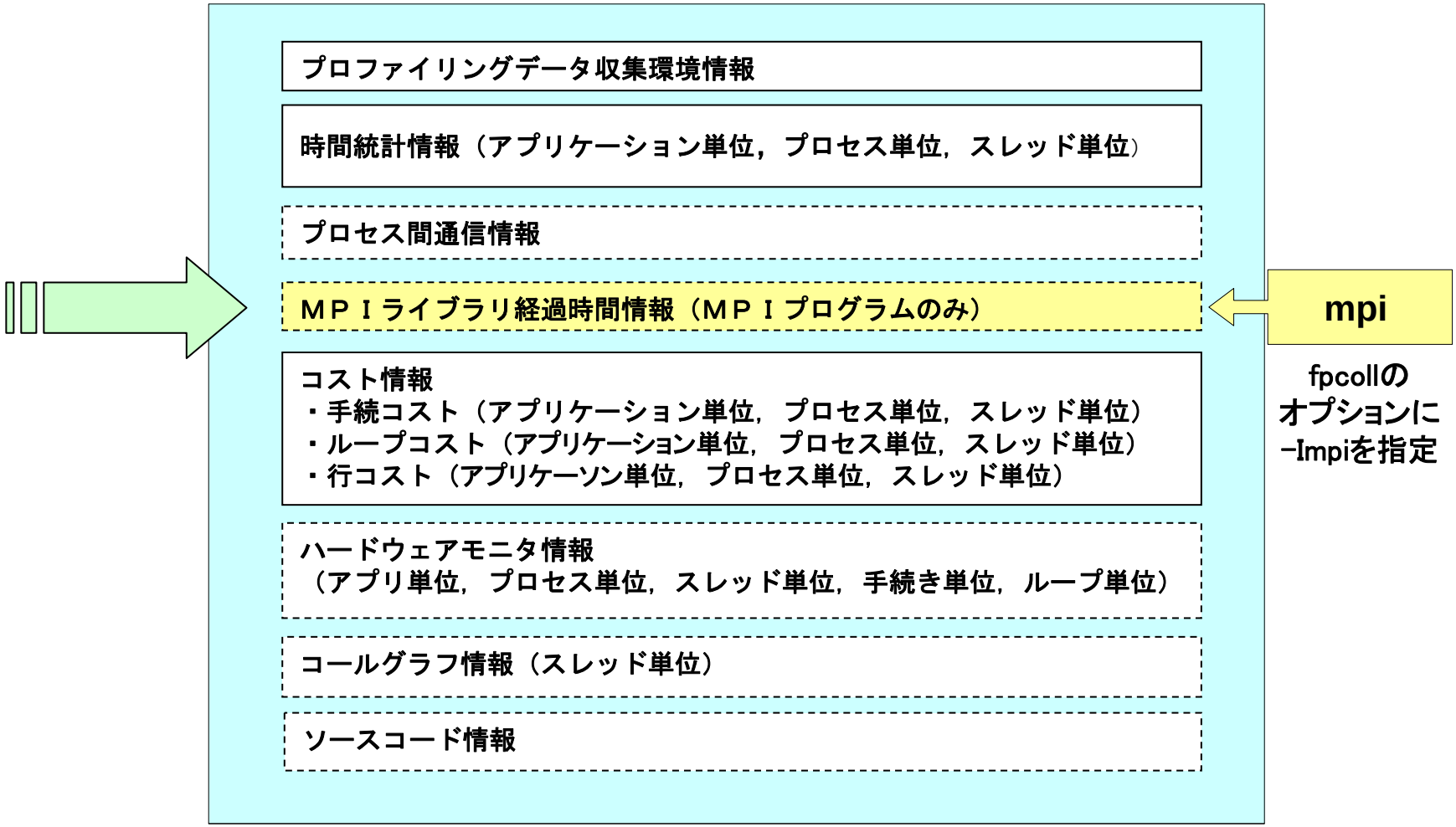
ソース翻訳時(オプション追加後)	プログラム実行時(オプション追加後)
pp do k=1,kmax ← kmaxの値は未定 pp do j=1,jmax ← jmaxの値は未定 pp do i=1,imax ← imaxの値は未定 ・・・ p end do p end do p end do **I, J, Kの軸から動的に選択**	pp do k=1,kmax ← **kmax=1** pp do j=1,jmax ← **jmax=100** pp do i=1,imax ← **imax=100** ・・・ p end do p end do p end do **Jの軸で並列実行**

6. MPI並列チューニング

6．1．MPI並列性能の評価（1）

JAXA Supercomputer System

MPI並列性能のホットスポット特定および分析を行うには、プロファイラ情報の以下の箇所に注目します。



6．2．MPI並列性能の評価(2)

JAXA Supercomputer System

■ MPIライブラリ経過時間情報・・・MPIライブラリ情報の抽出

MPI ライブラリ 経過時間

```
**********************************************************************************************
     Application - MP
**********************************************************************************************

    Elapsed(s)                Call to
-----------------------------------------------
      75.6673    ---.----     ------------   Application
-----------------------------------------------
       4.3298      5.7222         25920     MAIN__ (     2 -   1526)
       0.0642      0.0848           192     suba_  (  1712 -   1974)
       0.0106      0.0139           576     subb_  (  1975 -   2151)
       0.0000      0.0001           832     clock_ (  2152 -   2163)

    Elapsed(s)          %       Called by
-----------------------------------------------
       4.3298    ---.----     ------------   MAIN__
-----------------------------------------------
       1.8437     42.5816         23040     mpi_sendrecv_
       1.6820     38.8464          1536     mpi_allreduce_
       1.0006     23.1094            64     mpi_init_
       0.0639      1.4756          1024     mpi_barrier_
       0.0214      0.4953            64     mpi_finalize_
       0.0008      0.0191            64     mpi_gather_
       0.0000      0.0009            64     mpi_comm_rank_
       0.0000      0.0002            64     mpi_comm_size_
```

コスト比率

手続き名＋行番号のリスト

手続きから呼ばれる MPI 関数名

6．3．MPI並列性能の分析

採取したプロファイラ情報が以下の判定条件に該当する場合は、
対応するポイントの性能チューニングを行います。

プロファイラ結果の判定条件	MPI並列チューニングのポイント
MPI関数の呼出しコストが大きい	・プロセス間の通信コストを削減する
プロセス間のバランスが不均等なルーチンがある	・ロードバランスを均等化させる

6．4．MPI並列チューニング事例(1)

JAXA Supercomputer System

MPIのノンブロッキング通信の使用

MPI_SEND転送処理をノンブロッキング通信(MPI_ISEND)に変更することで通信コストを削減します。以下に書換え例を示します。

ソース変更前	ソース変更後

m = myrank + 1
if(m.le.npe-npl) then
mm = m+npl
call MPI_SEND(Buf1s,jmax1*lmax1*14*2,
& MPI_DOUBLE_PRECISION,mm-1,
& itag1,MPI_COMM_WORLD,IERR)
end if
m = myrank + 1
if(m.ge.1+npl) then
mm = m-npl
call MPI_RECV(Buf1r,jmax1*lmax1*14*2,
& MPI_DOUBLE_PRECISION,mm-1,
& itag1,MPI_COMM_WORLD,status,IERR)
	m = myrank + 1
	if(m.le.npe-npl) then
	mm = m+npl
	call MPI_ISEND(Buf1s,jmax1*lmax1*14*2,
	& MPI_DOUBLE_PRECISION,mm-1,
	& itag1,MPI_COMM_WORLD,ireq1s,IERR)
	end if
	m = myrank + 1
	if(m.ge.1+npl) then
	mm = m-npl
	call MPI_IRECV(Buf1r,jmax1*lmax1*14*2,
	& MPI_DOUBLE_PRECISION,mm-1,
	& itag1,MPI_COMM_WORLD,ireq1r,IERR)
	end if
	m = myrank + 1
	if(m.ge.1+npl) then
	call MPI_WAIT(ireq1r,status,IERR)
	end if
	m = myrank + 1
	if(m.le.npe-npl) then
	call MPI_WAIT(ireq1s,status,IERR)
	end if

6．5．MPI並列チューニング事例(2)

JAXA Supercomputer System

■ 通信回数の削減

複数の通信メッセージを統合して1回で送受信することにより、通信オーバーヘッドを削減します。以下に書換え例を示します。

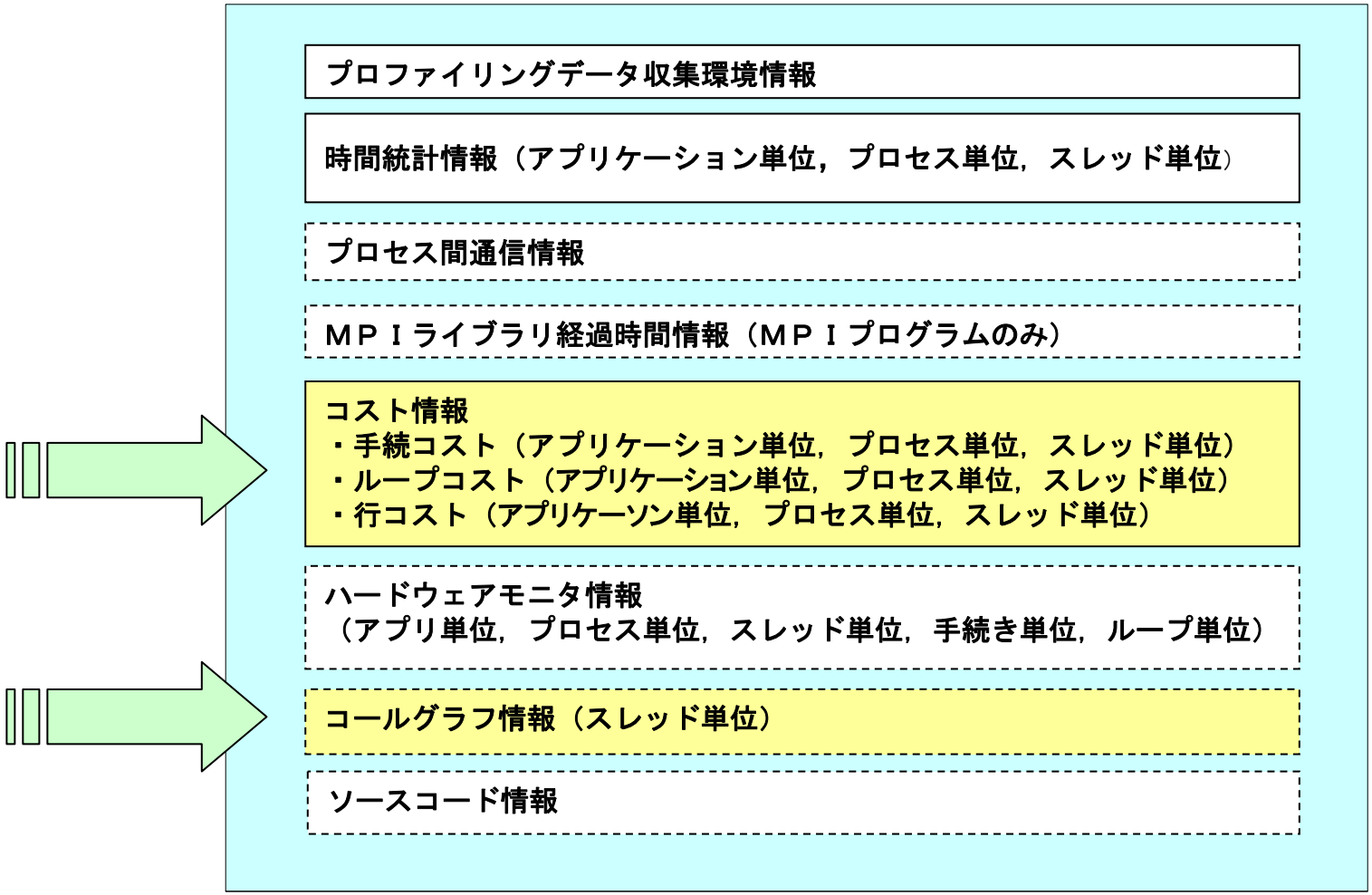
ソース変更前	ソース変更後

do l=1,nz
  do j=1,nx
  bufL(j,l)=a(j,1,l)
  end do
end do
call mpi_isend(bufL,nxz,MPI_REAL4,ilproc,iTag,MPI_COMM_WORLD,iLe,ierr)
・・・
do l=1,nz
  do j=1,nx
  bufL(j,l)=b(j,1,l)
  end do
end do
call mpi_isend(bufL,nxz,MPI_REAL4,ilproc,iTag,MPI_COMM_WORLD,iLe,ierr)
・・・
do l=1,nz
  do j=1,nx
  bufL(j,l)=c(j,1,l)
  end do
end do
call mpi_isend(bufL,nxz,MPI_REAL4,ilproc,iTag,MPI_COMM_WORLD,iLe,ierr)
・・・
	do l=1,nz
	  do j=1,nx
	  bufL(j,l,1)=a(j,1,l)
	  bufL(j,l,2)=b(j,1,l)
	  bufL(j,l,3)=c(j,1,l)
	  end do
	end do
	call mpi_isend(bufL,nxz*3,MPI_REAL4,ilproc,iTag,MPI_COMM_WORLD,iLe,ierr)
	・・・

7. XPFortran並列チューニング

7．1．XPFortran並列性能の評価（1）

XPFortran並列性能のホットスポット特定および分析を行うには、プロファイラ情報の以下の箇所に注目します。



7．2．XPFortran並列性能の評価（2）

JAXA Supercomputer System

■ コスト情報・・・XPFortranライブラリ情報の抽出

コスト

コスト比率

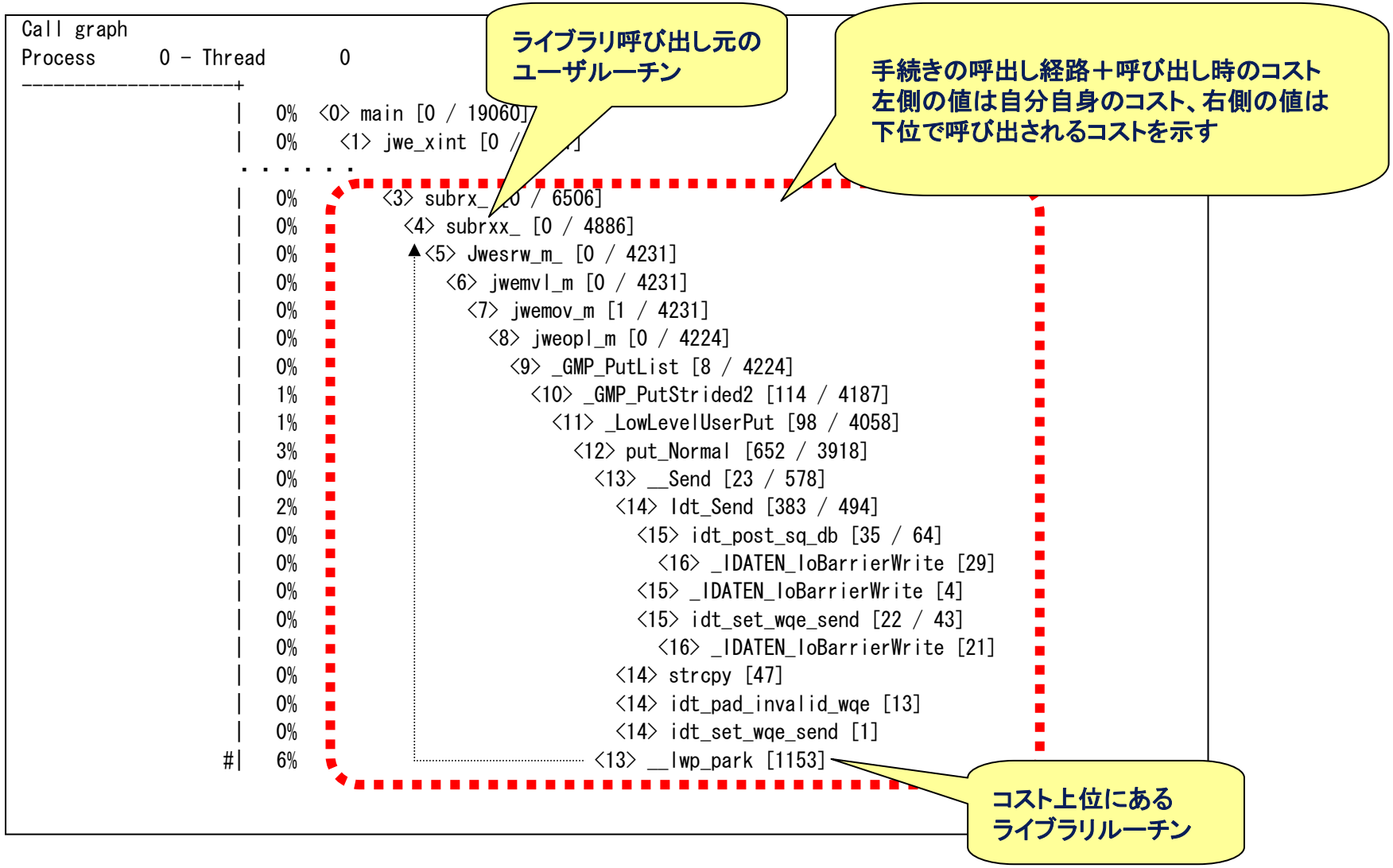
```
**************************  ****************************************
    Process       0 - proced
**************************  ********************************************************************

          Cost         %          Barrier          %     Start       End

-------------------------------------------------------------------------------------------------
         20130   100.0000               40     0.1987        --        --    Process       0
-------------------------------------------------------------------------------------------------
          2411    11.9771               32     1.3273        59       116    step_
          2162    10.7402                0     0.0000        --        --    __lwp_park
          1376     6.8356                0     0.0000        --        --    strcpy
          1272     6.3189                0     0.0000        --        --    isw_check_cqe
          1174     5.8321                0     0.0000        --        --    put_Normal
          1103     5.4794                0     0.0000        --        --    idt_purge_sq
          1041     5.1714                0     0.0000        --        --    lwp_yield
           874     4.3418                0     0.0000        --        --    lsw_BarrierProbe
           772     3.8351                0     0.0000        --        --    ldt_Send
           731     3.6314                0     0.0000        89        92    step._PRL_1_
```

手続き名＋行番号のリスト
ユーザ定義ルーチン以外の名前は
システム内部から呼び出されている
ライブラリであることを示す

7．3．XPFortran並列性能の評価(3)

JAXA Supercomputer System

■ コールグラフ情報・・・呼出し元ユーザルーチンの特定



7．4．XPFortran並列性能の分析

JAXA Supercomputer System

採取したプロファイラ情報が以下の判定条件に該当する場合は、
対応するポイントの性能チューニングを行います。

プロファイラ結果の判定条件	XPFortran並列チューニングのポイント
ユーザ定義以外のライブラリのコストが大きい	・プロセス間の通信コストを削減する
プロセス間のバランスが不均等なルーチンがある	・並列化率を上げる ・並列化粒度を上げる ・ロードバランスを均等化させる

7．5．XPFortran並列チューニング事例（1）

JAXA Supercomputer System

SPREAD DOループの融合

回転範囲および分割が共通のSPREAD DOループを融合することにより、オーバーヘッド削減による性能向上が期待できます。

ソース変更前

```
!XOCL SPREAD DO /IPL
          DO 130 L = 2, LM
           DO 130 K = 2, KM
                ・・・
            UXL( K,L,1 ) = VXL( K,L,1 )
            UXL( K,L,2 ) = VXL( K,L,2 )
            UXL( K,L,3 ) = VXL( K,L,3 )
 130      CONTINUE
!XOCL END SPREAD
!XOCL SPREAD DO /IPL
         DO 150 L = 2, LM
         DO 150 K = 2, KM
           SS   = 0.
                ・・・
           VXL( K,L,1 ) = SL( JS,K,L,1 )*FX1
           VXL( K,L,2 ) = SL( JS,K,L,2 )*FX1
           VXL( K,L,3 ) = SL( JS,K,L,3 )*FX1
 150      CONTINUE
!XOCL END SPREAD
```

ソース変更後

```
!XOCL SPREAD DO /IPL
           DO 130 L = 2, LM
            DO 130 K = 2, KM
                 ・・・
             UXL( K,L,1 ) = VXL( K,L,1 )
             UXL( K,L,2 ) = VXL( K,L,2 )
             UXL( K,L,3 ) = VXL( K,L,3 )
            SS   = 0.
                 ・・・
            VXL( K,L,1 ) = SL( JS,K,L,1 )*FX1
            VXL( K,L,2 ) = SL( JS,K,L,2 )*FX1
            VXL( K,L,3 ) = SL( JS,K,L,3 )*FX1
  130     CONTINUE
!XOCL END SPREAD
```

ループの内容によっては、融合するとかえって性能低下する場合もあります。

7．6．XPFortran並列チューニング事例(2)

JAXA Supercomputer System

■ 計算と通信のオーバーラップ

通信処理のコストが大きく、周囲に通信と独立して実行可能な計算処理が存在する場合、通信と計算を同時に行わせることで、実行時間を短縮することができます。

ソース変更前

```
!XOCL SPREAD MOVE /IPK
      DO 800 K = 1, KMAX
      DO 800 L = 1, LMAX
      DO 800 J = 1, JMAX
        S(J,K,L,1) = SK(J,K,L,1)
        S(J,K,L,2) = SK(J,K,L,2)
         :
800 CONTINUE
!XOCL END SPREAD (MOV4)
!XOCL MOVEWAIT (MOV4)
!XOCL SPREAD DO /IPL
      DO 40 L = 2, LM
      DO 40 K = 2, KM
      DO 40 J = 2, JM
       RJ          =1./Q(J,K,L,6)
       SL(J,K,L,2)=SL(J,K,L,2)-Q(J,K,L,3)*RJ
       SL(J,K,L,3)=SL(J,K,L,3)+Q(J,K,L,2)*RJ
   40 CONTINUE
!XOCL END SPREAD
```

ソース変更後

```
!XOCL SPREAD MOVE /IPK
      DO 800 K = 1, KMAX
      DO 800 L = 1, LMAX
      DO 800 J = 1, JMAX
        S(J,K,L,1) = SK(J,K,L,1)
        S(J,K,L,2) = SK(J,K,L,2)
         :
800 CONTINUE
!XOCL END SPREAD (MOV4)
!XOCL SPREAD DO /IPL
      DO 40 L = 2, LM
      DO 40 K = 2, KM
      DO 40 J = 2, JM
       RJ          =1./Q(J,K,L,6)
       SL(J,K,L,2)=SL(J,K,L,2)-Q(J,K,L,3)*RJ
       SL(J,K,L,3)=SL(J,K,L,3)+Q(J,K,L,2)*RJ
   40 CONTINUE
!XOCL END SPREAD
!XOCL MOVEWAIT (MOV4)
```

まとめ

JAXA Supercomputer System

■ 性能チューニングのポイント

スカラチューニング

- ◆データの局所性を高める
- ◆演算器の実行効率を高める

MPI並列チューニング

- ◆ロードバランスを均等化させる
- ◆プロセス間の通信コストを削減する

自動並列チューニング

- ◆並列化率を上げる
- ◆並列化粒度を上げる
- ◆ロードバランスを均等化させる

XPFortran並列チューニング

- ◆並列化率を上げる
- ◆並列化粒度を上げる
- ◆ロードバランスを均等化させる
- ◆プロセス間の通信コストを削減する

■ 性能チューニングの目安

性能指標	性能目安
MIPS値（プロセスあたり）	4000以上
MFLOPS値（プロセスあたり）	2000以上
L2キャッシュミス率（プロセスあたり）	0.2%未満

目安を満たしていない場合は、本書の性能チューニングをご検討ください。