インテル® アーキテクチャー向け 最適化メソッドのご紹介

エクセルソフト株式会社

黒澤 一平

Code the Future

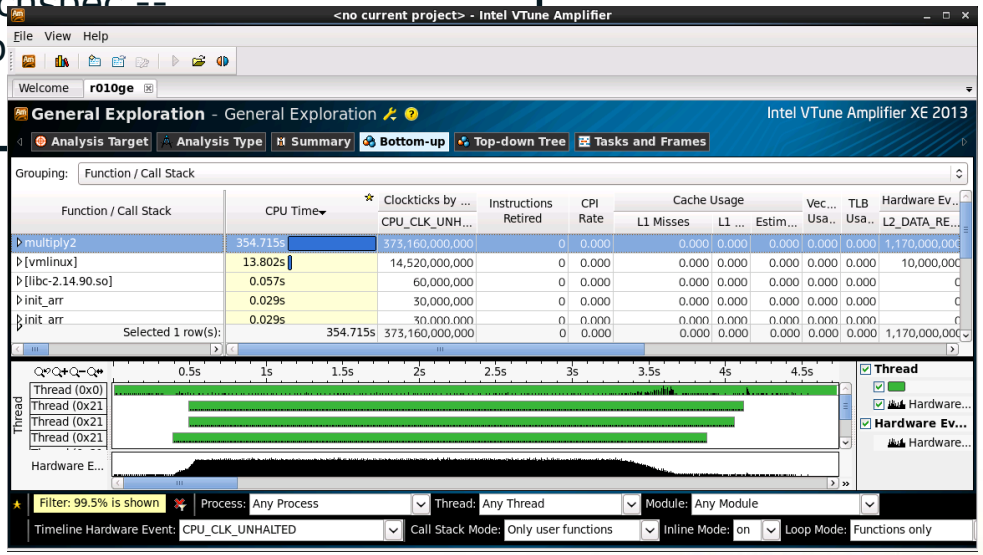
インテル® VTune™ Amplifier XE の概要

- パフォーマンス・ボトルネック解析プロファイラー
  - プログラムの実行に時間がかかる処理、足かせとなる箇所を特定する
- 動的サンプリングで統計データを表示
  - 実行中のターゲットに対し、システム割り込みを利用して各種データを収集する
- アプリケーション・レベルのパフォーマンス解析
  - Hotspots（CPU 使用率の高い処理）の検出が可能
- マイクロアーキテクチャー・ レベルのパフォーマンス解析
  - CPI 値、キャッシュミス、メモリー帯域幅、分岐予測ミス、ストールなどの解析が可能

Code the Future

インテル® VTune™ Amplifier XE のコマンドラインと GUI

```
$ amplxe-cl -collect knc-hotspots -search-dir
all=/lib/firmware/mic -search-dir
all=/home/spec/benchspec --
/home/spec/benchsp
```



サーバーや、クラスターシステムなどの GUI 環境のないシステムでは、コマンドラインでリモート解析し、GUI 環境のあるシステムで結果を確認することができる

Code the Future

OPTIMIZATION NOTICE

サンプリング手法

intel Software

◆ ユーザーモード・サンプリングおよびトレース・コレクション（User-Mode Sampling and Tracing Collection）または “タイムベース・サンプリング・コレクション”(TBS)

MIC 非対応

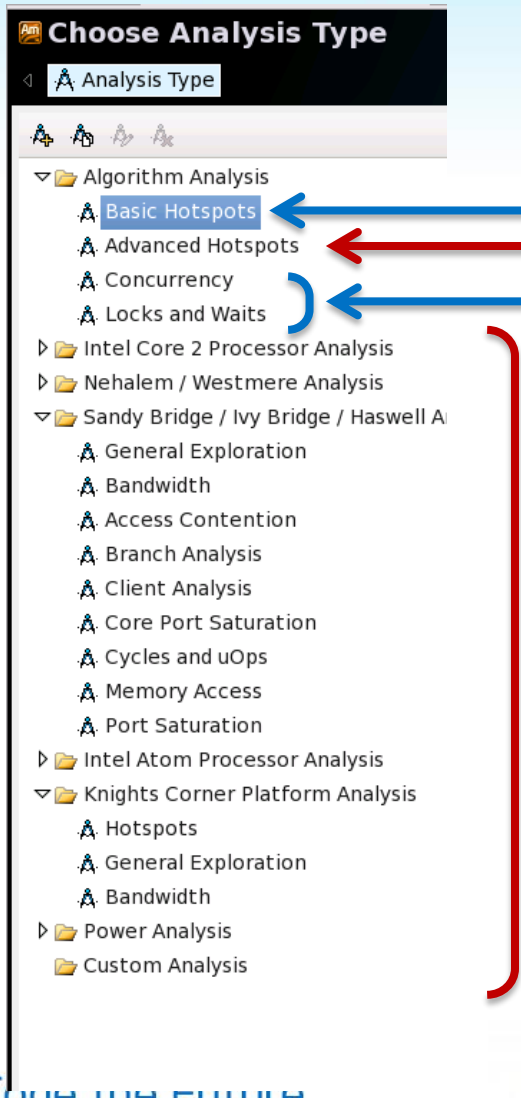
- プロセッサーに周期的に割り込みを入れて、その割り込み処理の中で、IP (インストラクション・ポインター) や各種スレッド情報などを収集するサンプリング手法。
- デフォルトでは、10ms に 1 回の割合で割り込みを入れてデータを収集する。
- その際のオーバーヘッドは約 5%。

◆ イベントベース・サンプリング・コレクション (EBS)（Hardware Event-based Sampling Collection）

MIC 対応

- インテル® プロセッサーに搭載されるパフォーマンス・モニタリング・ユニット (PMU) のイベントカウンター・オーバーフローによる割り込みを利用したサンプリング手法。
- この割り込みの発生回数がサンプリング回数となる。
- 割り込み発生の頻度は、PMU のオーバーフロー値を決定する “Sample After Value”を調節することで制御できる。
- 1ms 単位で割り込みが発生した場合、それによるオーバーヘッドは約 2% 程度。
- 本サンプリング機能を使用できるプロセッサーは、以下のように限定される。
  サポート対象 CPU: インテル® Pentium® M プロセッサー、インテル® Core™ マイクロアーキテクチャー以降
  インテル® Atom™ プロセッサー
  サポート非対象 CPU: インテル® Pentium® 4 プロセッサー・ファミリー以前、インテル以外のプロセッサー
- NMI Watchdog の設定は無効にする。

プリセットされた解析タイプ



イベントベース・サンプリング（EBS）

タイムベース・サンプリング（TBS）

イベントベース・サンプリング（EBS）

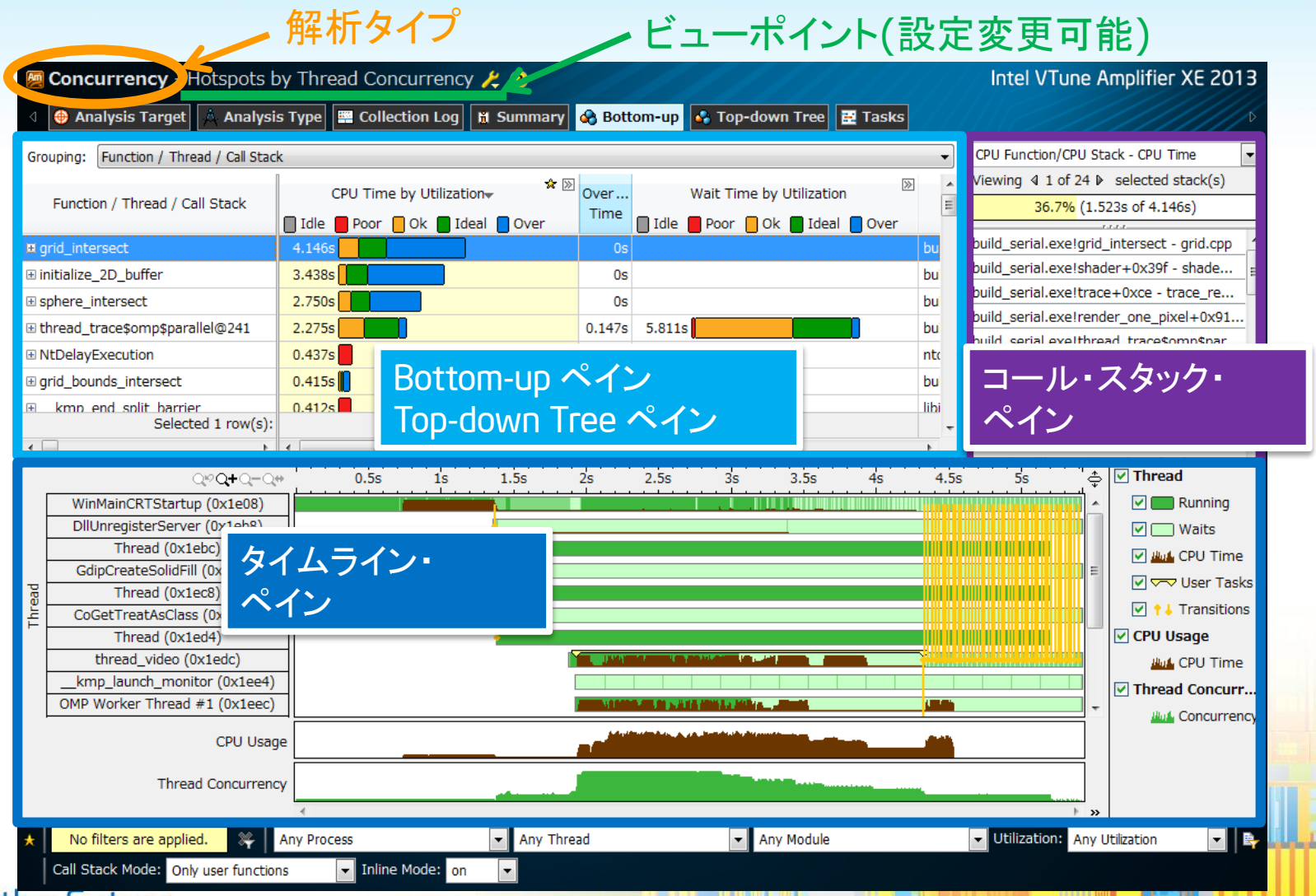
解析タイプ (コマンドライン)

<EBS>
advanced-hotspots
snb-general-exploration
knc-hotspots
knc-general-explorationなど

<TBS>
hotspots
concurrency
locksandwaits

Code the Future

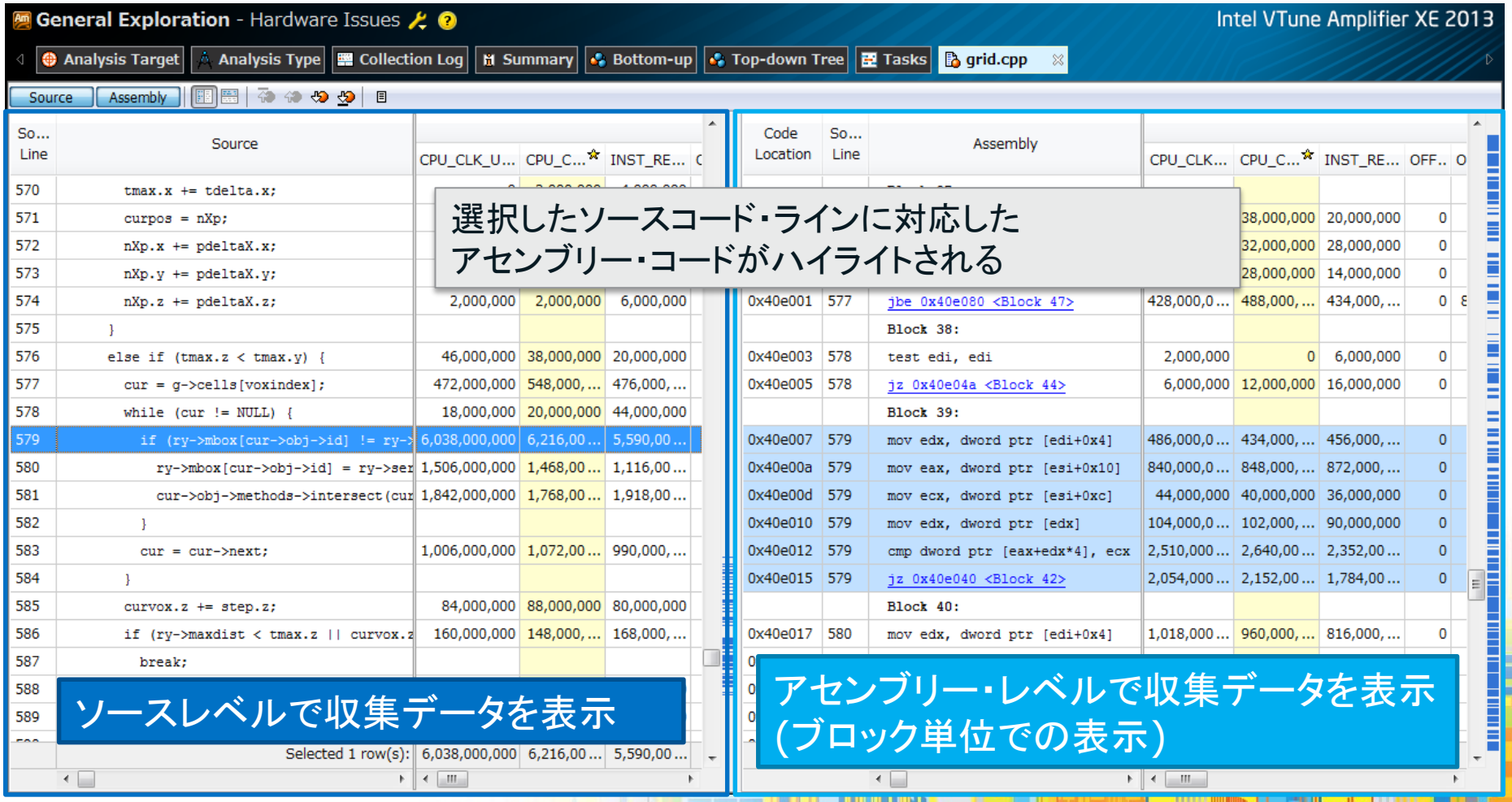
結果内容（GUI）



Code the Future

ソースコードの表示

- 関数レベルからソースコード・レベルへドリルダウンして結果を表示



OPTIMIZATION NOTICE

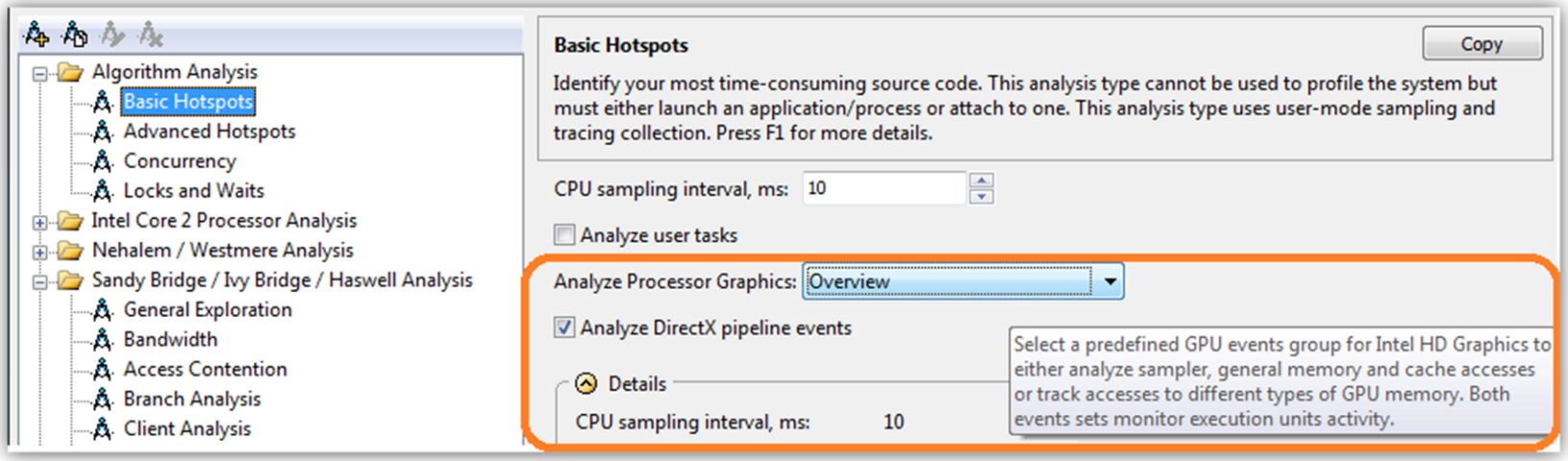
結果内容（コマンドライン）

```
Collection and Platform Info
----------------------------
Parameter                   r001bw
--------------------------  ----------------------------------------
Application Command Line    C:\Test\MyApp.exe
Computer Name               My Computer
Environment Variables
MPI Process Rank
Operating System            Microsoft Windows 7
Result Size                 7167211
User Name

CPU
---
Parameter           r001bw
------------------  ----------------------------------------
Frequency           2600000000
Logical CPU Count   4
Name                Intel(R) Core(TM) Processor 2xxx Series

Summary
-------
Elapsed Time:  9.367
CPU Usage:     0.023

Event summary
-------------
Hardware Event Type                                    Hardware Event Count:Self  Hardware Event Sample Count:Self  Even
-----------------------------------------------------  -------------------------  --------------------------------  ----
CPU_CLK_UNHALTED.THREAD                                1158000000                 579                               2000
CPU_CLK_UNHALTED.REF_TSC                               1248000000                 624                               2000
INST_RETIRED.ANY                                       688000000                  344                               2000
OFFCORE_RESPONSE.ANY_REQUEST.LLC_MISS_LOCAL.DRAM_0     6300000                    63                                1000
OFFCORE_RESPONSE.ANY_REQUEST.LLC_MISS_LOCAL.DRAM_1     6300000                    63                                1000
MEM_LOAD_UOPS_MISC_RETIRED.LLC_MISS_PS                 970000                     97                                1000

Uncore Event summary
--------------------
Hardware Event Type  Hardware Event Count:Self
-------------------  --------------------------
UNC_IMC_GT_REQUESTS  31635126
UNC_IMC_IA_REQUESTS  124751817
UNC_IMC_IO_REQUESTS  36705657
UNC_IMC_DATA_READS   148408289
UNC_IMC_DATA_WRITES  45161425
```

```
$ amplxe-cl -R summary -r r001ge

General Exploration Metrics
---------------------------
Parameter                                r001ge
---------------------------------------  -----------
Hardware Event Count
 CPU_CLK_UNHALTED.CORE                   16219170376
 INST_RETIRED.ANY                        710657200
CPI Rate                                 22.823
Retiring uOps                            0.024
 MS Assists                              0.006
 FP Assists                              0.0
 SIMD Assists                            0.0
Wasted Work                              0.0
 Branch Mispredict                       0.030
 SMC Machine Clear                       0.0
Back-end Issues                          0.0
 Memory Latency                          0.0
  LLC Miss                               0.124
  LLC Hit                                0.0
  DTLB Overhead                          0.0
  Contested Accesses                     0.0
  Page Walk                              0.022
  Bus Lock                               0.0
 Memory Reissues                         0.0
  Split Loads                            0.0
  Split Stores                           0.0
  Loads Blocked by Store Forwarding      0.0
 DIV Active                              0.0
Front-end Issues                         0.0
 ICache Misses                           0.0
 ITLB Overhead                           0.007
```

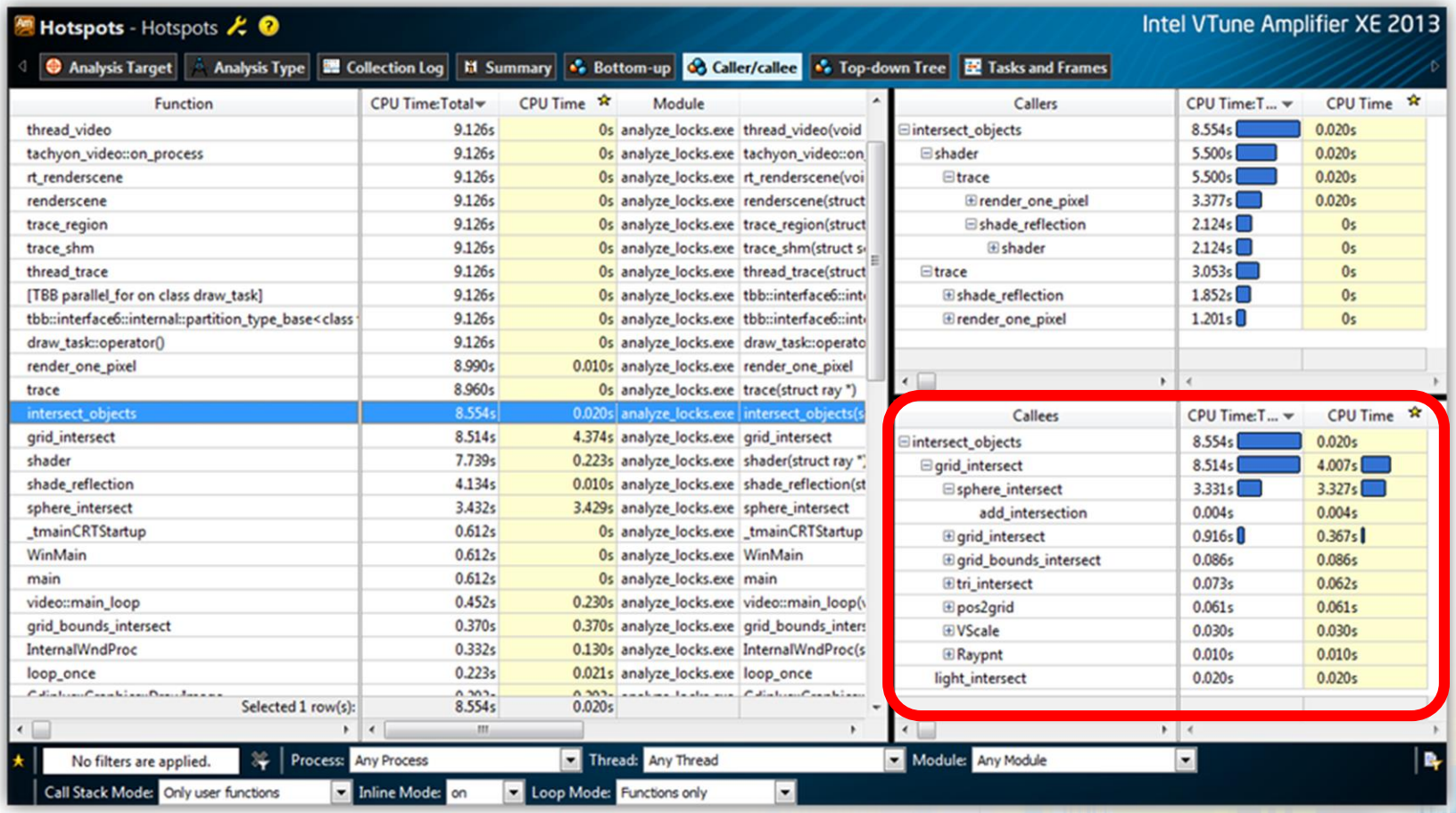
Code the Future

新機能：インテル® HD グラフィックス対応



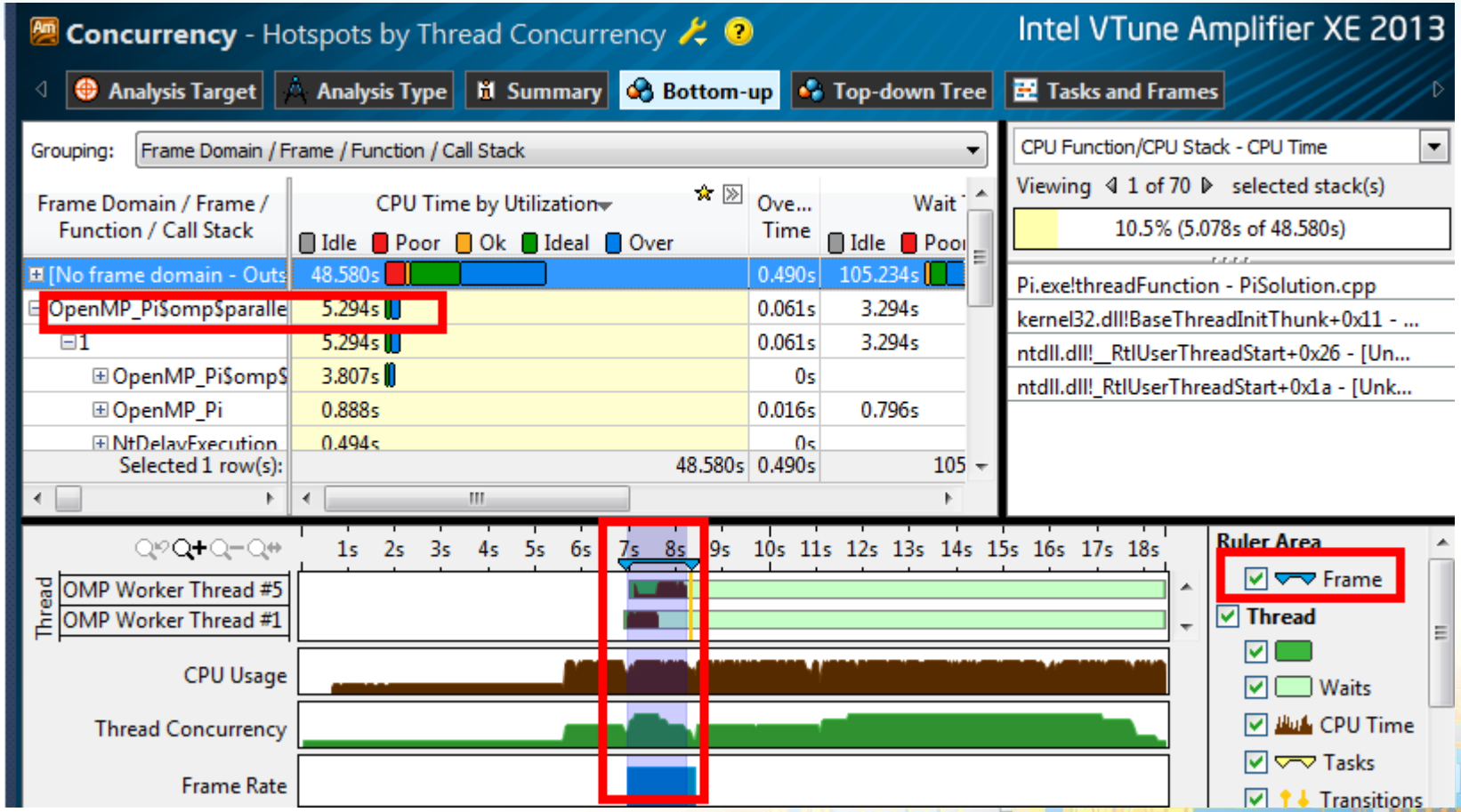
Code the Future

新機能：コーリー・スタック(呼び出し先情報)表示の追加



Code the Future

新機能：OpenMP* フレームの表示

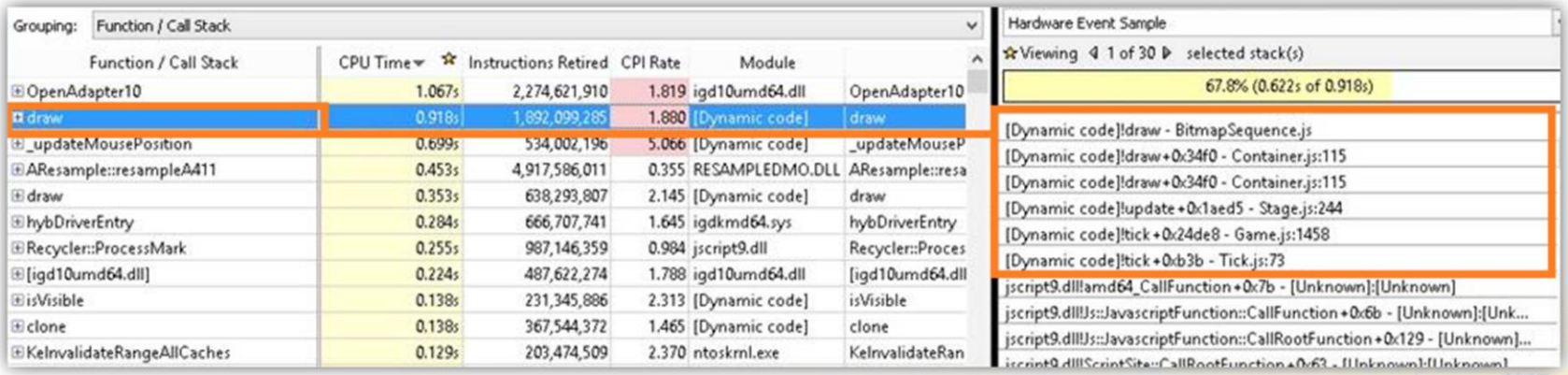


Code the Future

新機能: その他

- Java プロセスのハードウェア・イベントベース・サンプリングに対応

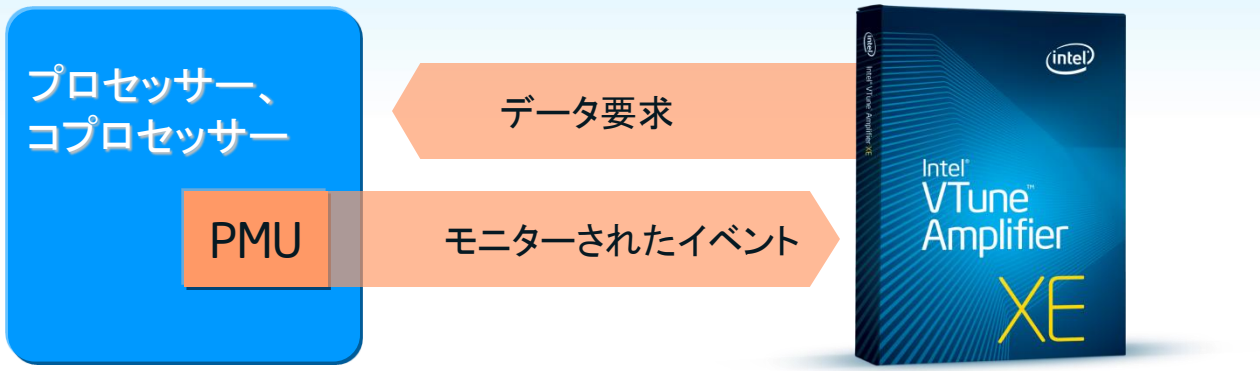
- Windows 8* において、C# と JavaScript ベースの Windows* ストア アプリケーションのハードウェア・イベントベース・サンプリングに対応



JavaScript の解析結果

Code the Future

Performance Monitoring Unit (PMU)



- プロセッサー、コプロセッサーに実装されている Performance Monitoring Unit (PMU) というバッファーに、モニターされたハードウェア・イベントが保存されている
- インテル® VTune™ Amplifier XE は設定したタイミングで PMU からデータを取得する
- 各コアから得た情報を統計することでイベントベース・サンプリング (EBS) の結果を表示する

Code the Future

インテル® VTune™ Amplifier XE のコマンドライン
※MPI アプリケーションの解析はコマンドラインのみ対応

amplxe-cl -collect <*Analysis Type*> -- <*program*>

<*Analysis Type*>
lightweight-hotspots、hotspots、concurrency、locksandwaits、
core2-general-exploration、nehalem-general-exploration、
snb-general-exploration
など

コマンド例:
amplxe-cl -collect snb-general-exploration -- ./sample.exe

MPI アプリケーションの場合は 1 ノードに 1 起動するように設定

mpiexec -host host001 -n 1 amplxe-cl -collect <Analysis Type> -r <name> -- <program>: -host host001 -n 15 <program>: -host host 002 -n 1 amplxe-cl -collect <Analysis Type> -r <name> -- <program>: -host host002 -n 15 ...

Code the Future

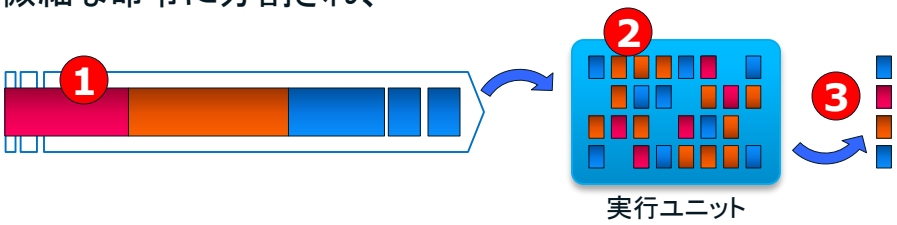
CPI Rate (Clockticks per Instructions Retired Rate)

CPI Rate = Clockticks / Instructions Retired
= CPU_CLK_UNHALTED.THREAD / INST_RETIRED.ANY

ハードウェアが効果的に活用できているかを確認することができます。インテル® Core™ i7 プロセッサーやインテル® Xeon® プロセッサーの場合には 0.25 が最良値であり、この値のセルが赤くなる場合には最適化によりパフォーマンスを考える必要があります。

CPI Rate の最良値が 0.25 である理由

1. 命令セットはマイクロオペレーションと呼ばれる微細な命令に分割され、

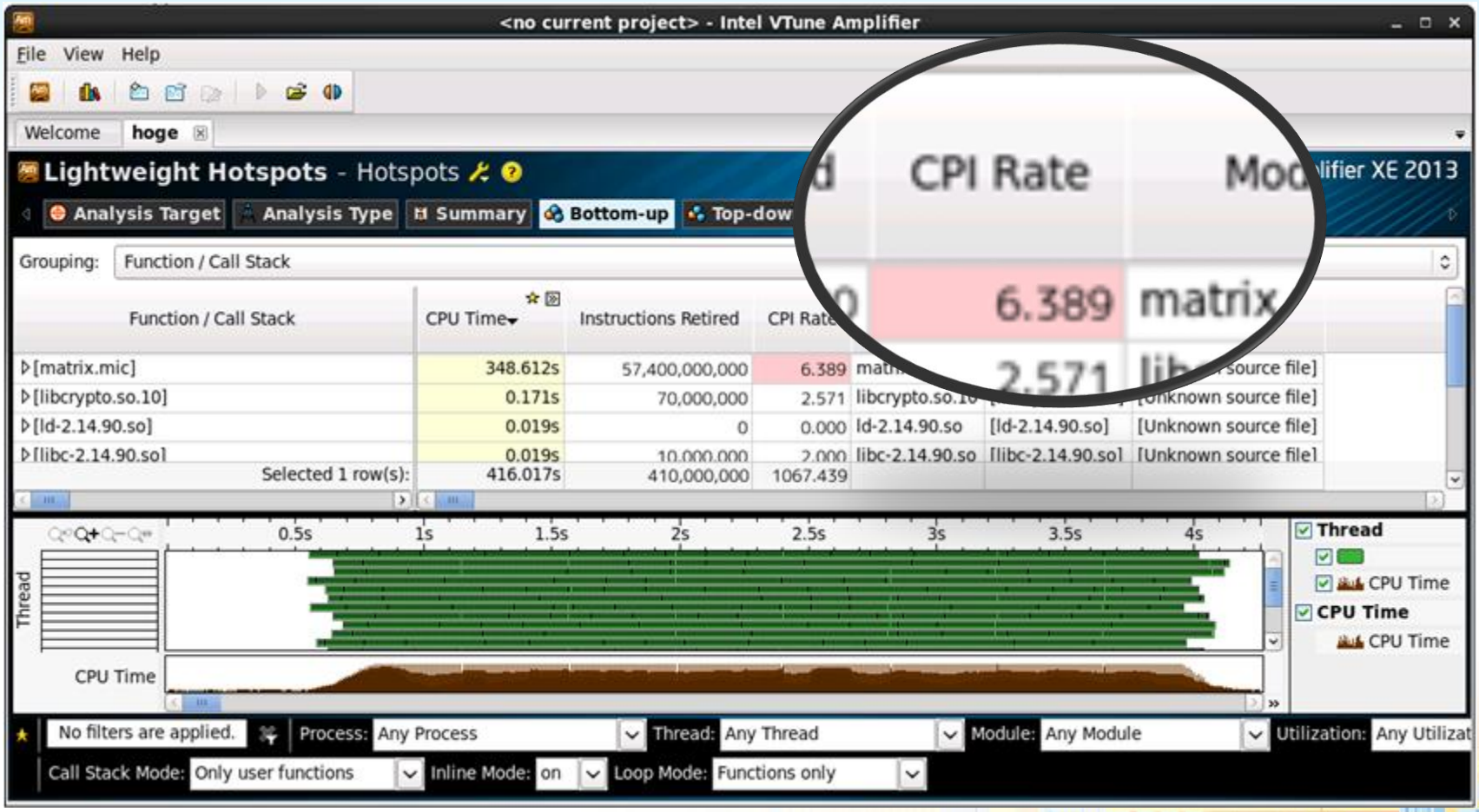


2. コア内では、実行可能なマイクロオペレーションから先に順次実行されます。

3. インテル® Core™ i7 プロセッサーでは、1 コア内で最大 4 つのマイクロオペレーションを同時に完了させることができます。

1 CPU サイクルで 4 つのマイクロオペレーションを完了させた場合、CPI Rate は 1/4＝0.25 となります。
CPI Rate により、ハードウェアの性能を正しく発揮することができたか確認することができます。

CPI を確認



この関数は CPI=6.389 であるため、さらなる性能の向上が期待できる

Code the Future

インテル® Xeon Phi™ コプロセッサーの CPI

インテル® Xeon Phi™ コプロセッサーのスレッドあたりの CPI
= CPU_CLK_UNHALTED / INSTRUCTIONS_EXECUTED

1 命令あたりのサイクル数、CPI を確認することで効率性を評価することができる

CPI は小さいほど効率性が高いことを意味する

最小理論値はアーキテクチャーによって異なる
・インテル® Xeon® プロセッサーは CPI=0.25 が最良であったが、
・インテル® Xeon Phi™ コプロセッサーは CPI=2 が最良

インテル® Xeon Phi™ コプロセッサーでは、CPI=4 以上は非効率

イベント	意味
CPU_CLK_UNHALTED	実行されたコアのサイクル数
INSTRUCTIONS_EXECUTED	スレッドによって実行された命令の数

インテル® VTune™ Amplifier XE を使用したパフォーマンス分析の一般的な手順

1. Hotspots（CPU 時間を要している関数）を特定

2. イベントベース・サンプリングを行い Hotspots の効率性を評価する

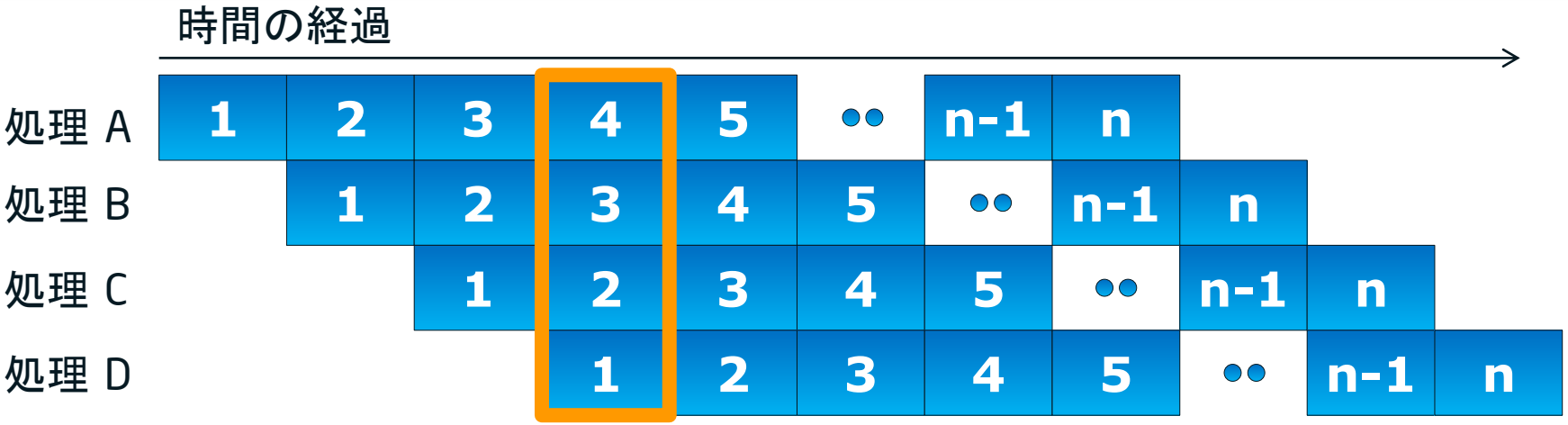
3. Hotspots が非効率的と判断できる場合、パフォーマンスに影響を及ぼすハードウェア・イベントを確認し、問題を修正する

4. すべての重要な Hotspots が評価されるまで 1～3 を繰り返す

Code the Future

プロセッサーのパイプライン

- パイプラインと呼ばれる多段階の小さな処理ステージに分けられて実行される
- 複数の処理を隙間なく送り込むことによって、効率化されている



※ しかし、分岐予測ミスが生じるとこの中身をすべて入れ替えなければならないため、パイプラインが深いほど大きな遅延が発生する

最もパイプラインが深い Intel NetBurst® マイクロアーキテクチャー

パイプラインにおける分岐命令の影響

```
add  eax, 1
jz     ecx, label
mov mem, eax
mov mem, ecx
```

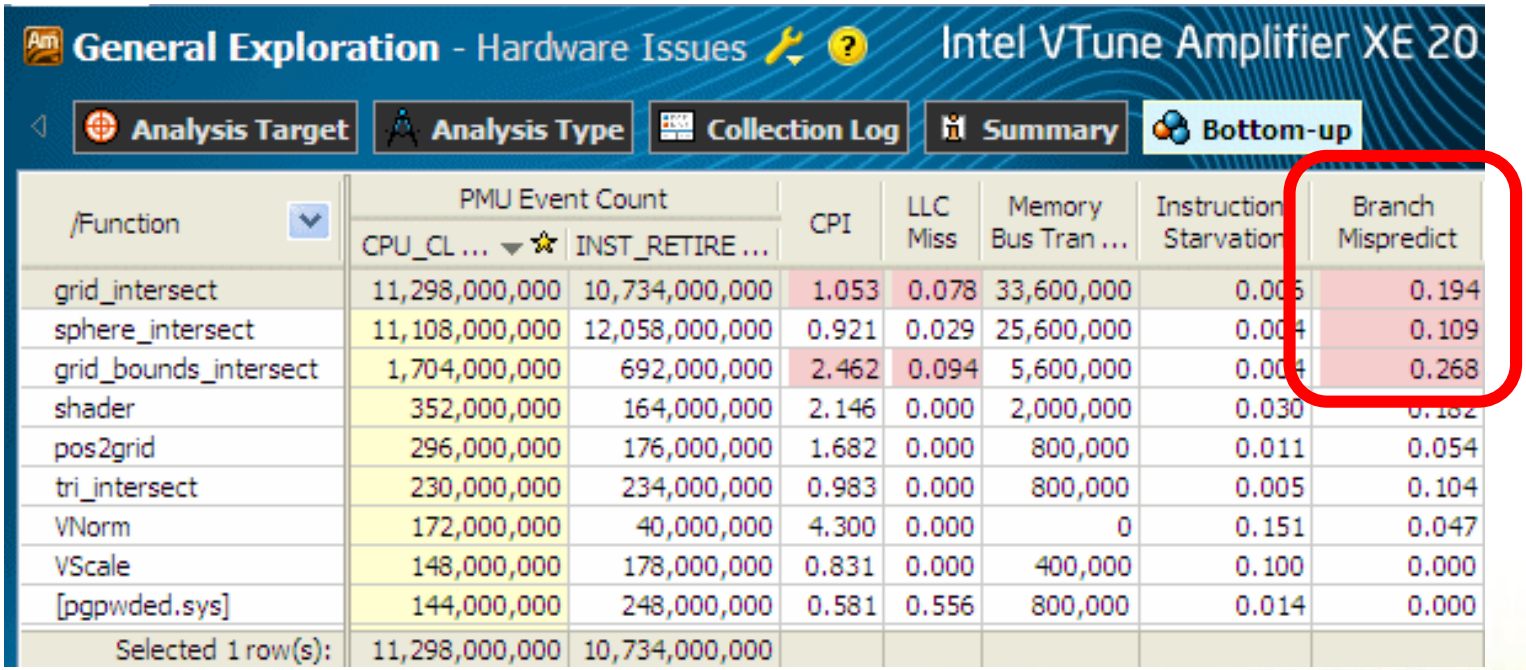
分岐命令はパイプラインの流れを変える

フェッチ	デコード	エグゼキューション	ライトバック
add			
jz	add		
mov	jz	add	
mov	mov	jz	add

先行する条件分岐命令が分岐すると、
これらの命令は廃棄される

パイプラインの段数が深いほど、分岐先の命令が実行されるまで時間がかかる

インテル® VTune™ Amplifier XE による分岐予測ミスの分析



/Function	PMU Event Count		CPI	LLC Miss	Memory Bus Tran ...	Instruction Starvation	Branch Mispredict
	CPU_CL ...	INST_RETIRE ...					
grid_intersect	11,298,000,000	10,734,000,000	1.053	0.078	33,600,000	0.006	0.194
sphere_intersect	11,108,000,000	12,058,000,000	0.921	0.029	25,600,000	0.004	0.109
grid_bounds_intersect	1,704,000,000	692,000,000	2.462	0.094	5,600,000	0.004	0.268
shader	352,000,000	164,000,000	2.146	0.000	2,000,000	0.030	0.182
pos2grid	296,000,000	176,000,000	1.682	0.000	800,000	0.011	0.054
tri_intersect	230,000,000	234,000,000	0.983	0.000	800,000	0.005	0.104
VNorm	172,000,000	40,000,000	4.300	0.000	0	0.151	0.047
VScale	148,000,000	178,000,000	0.831	0.000	400,000	0.100	0.000
[pgpwded.sys]	144,000,000	248,000,000	0.581	0.556	800,000	0.014	0.000
Selected 1 row(s):	11,298,000,000	10,734,000,000					

BR_INST_RETIRED.MISPRED / BR_INST_RETIRED.ANY
最良値は 0

Code the Future

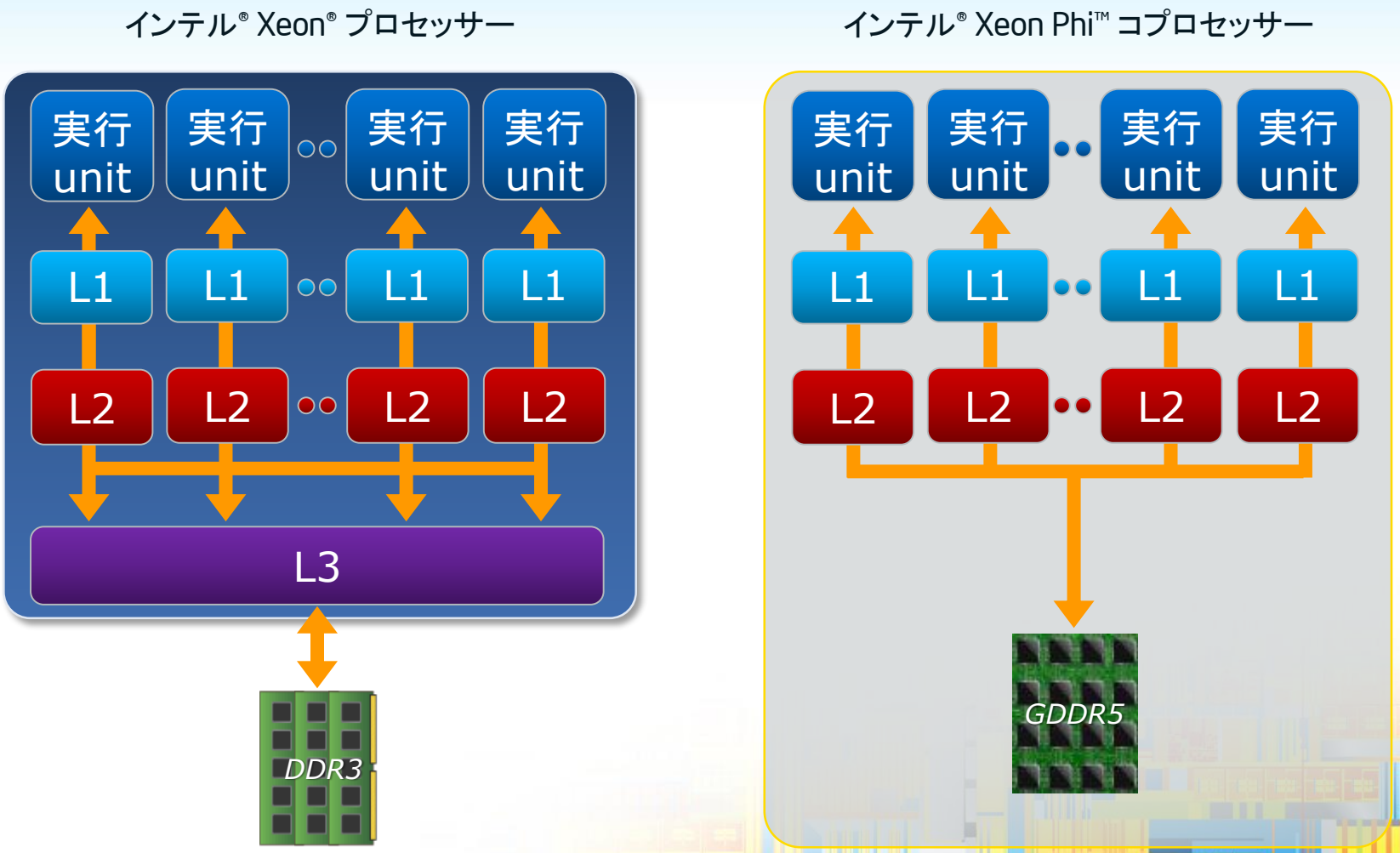
分岐予測ミスの回避

```
for (j = 0; j < size; j++ )
{
  if ( blend == 255 )
    dest[j] = src_1[j];
  else if ( blend == 0 )
    dest[j] = src_2[j];
  else
    dest[j] = ( src_1[j] * blend + src_2[j] *
(255-blend) ) / 256;
 }
```

```
if ( blend = 255 )
for ( j = 0; j < size; j++ )
  dest[j] = src_1[j];
else if ( blend == 0 )
for ( j = 0; j < size; j++ )
  dest[j]= src_2[j];
else
for ( j = 0; j < size; j++ )
  dest[j] = ( src_1[j] * blend + src_2[j] *
(255-blend) ) / 256;
```

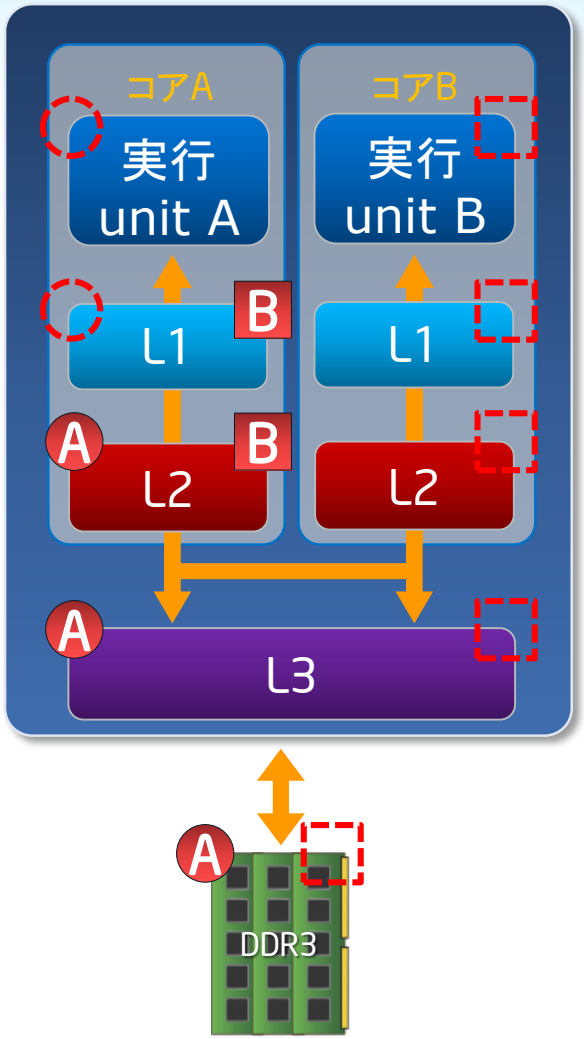
ループ内で評価不要な分岐をループの外に出して分岐予測ミスの発生をふせぐ

インテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーのキャッシュ構造の違い



Code the Future

キャッシュによる高速化

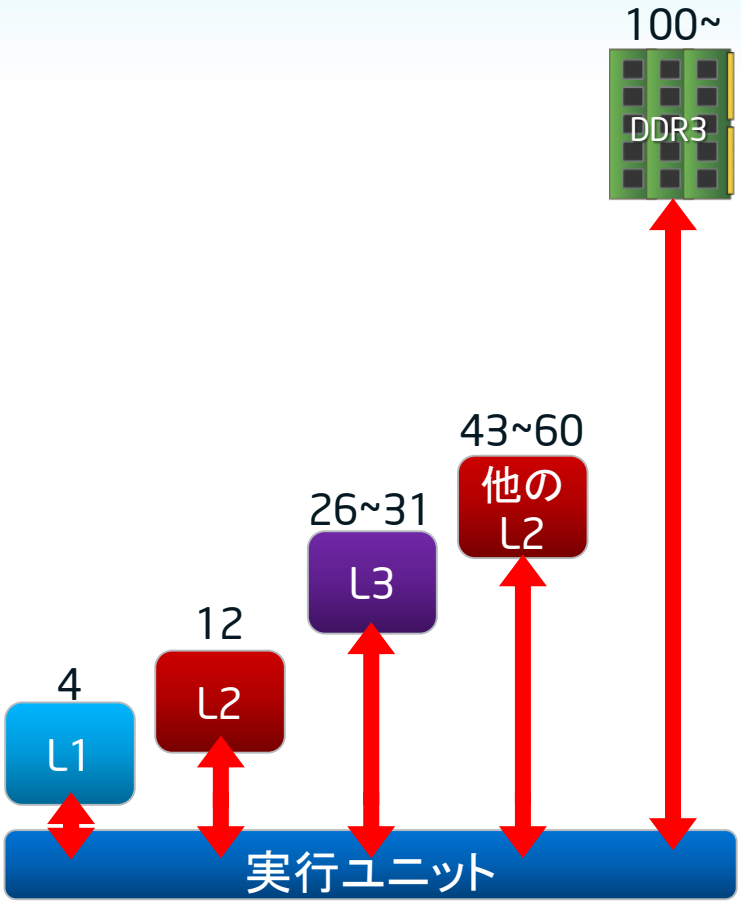


1. コアA がデータ A を要求
2. コアA の L2 キャッシュにコピーされていたデータ A が使用される

a. コアB がデータ B を要求
b. コアA が作り出したデータ B はコアA の L2 キャッシュから使用される

より近くから (時間のかからないキャッシュから) データを取り寄せるようにして、読み込み待ちの遅延を少なくしている

実行ユニットへのデータ転送に要する時間と容量



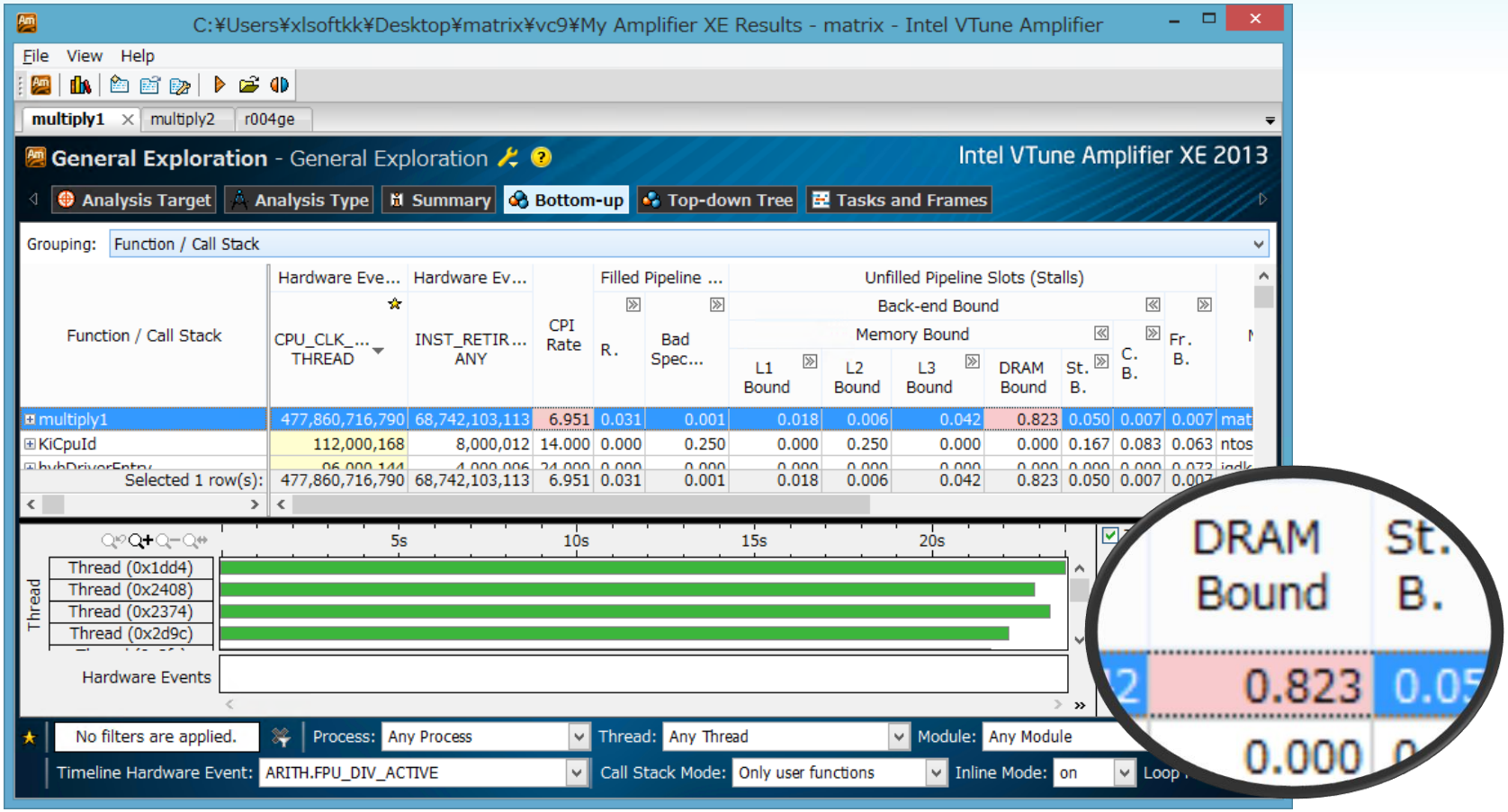
SandyBridge† アーキテクチャーの
転送時間 (単位は CPU サイクル)

	インテル® Xeon® プロセッサー	インテル® Xeon Phi™ コプロセッサー
L1	32KB	32KB
L2	256KB	512KB
L3	2MB/コア	なし
DDR3 / GDDR5	~4TB	~16GB

キャッシュ、メモリーの容量

†開発コード名

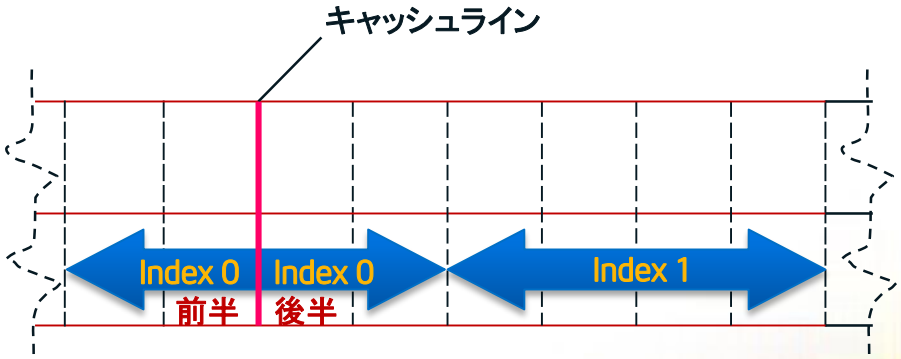
インテル® VTune™ Amplifier XE による データ転送によるパフォーマンス問題の分析



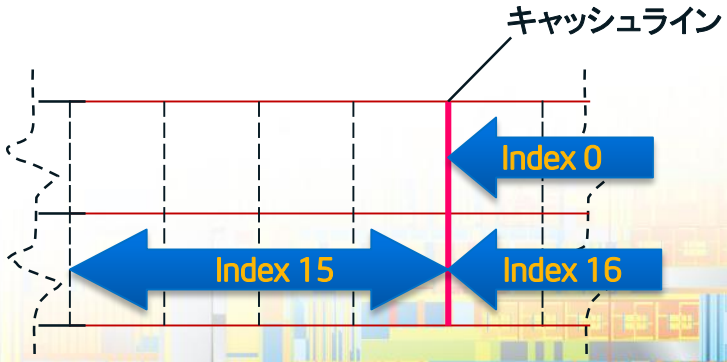
キャッシュミスが多発する箇所を特定

キャッシュラインの分割

- キャッシュラインは 64 バイト
- データ要素がキャッシュ境界をまたがないようにする
- キャッシュ境界をまたぐと、そのデータ要素にアクセスするためには、1 回ではなく 2 回のアクセスが必要になるため、パフォーマンスが低下する



キャッシュライン分割の状態



キャッシュラインをまたがない状態

Code the Future

アライメント

アライメントされているとより高速に処理することができる。
キャッシュラインの分割と同様に、高速演算のためには
16,32,64 バイト境界をまたがない配列の確保が有効。

インテル® AVX 向けには 32 バイト、インテル® MIC アーキテクチャー向けには 64 バイト (512bit) でアライメント

(C/C++)

```
__declspec(align(64)) float A[100][1024];
```

※ __ はアンダーバー2つ

(Fortran)

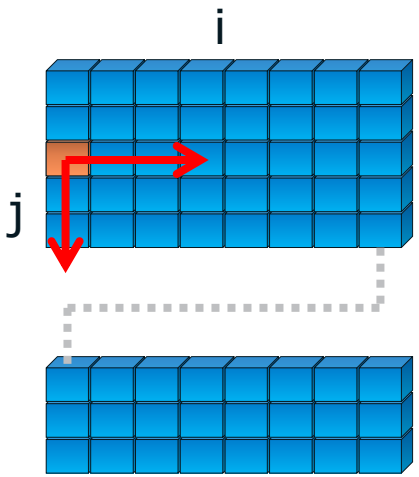
```
real*4 A(1024,100)
!DEC$ATTRIBUTES ALIGN: 64:: A
```

キャッシュの利用効率

コードの書き方で変わるキャッシュの利用効率

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

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

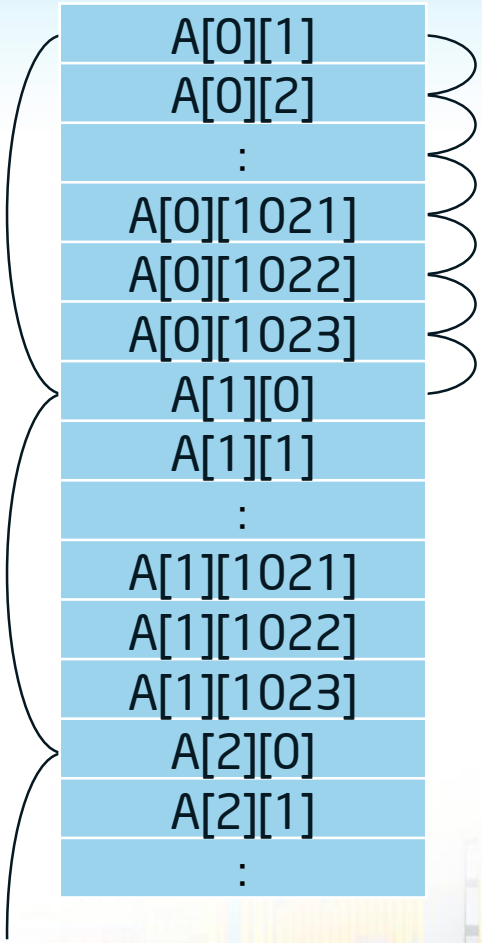


Code the Future

アクセスの効率性

非効率なアクセス

```
for (j=0; j<N; j++)
        A[j][i]
```



効率が良いアクセス

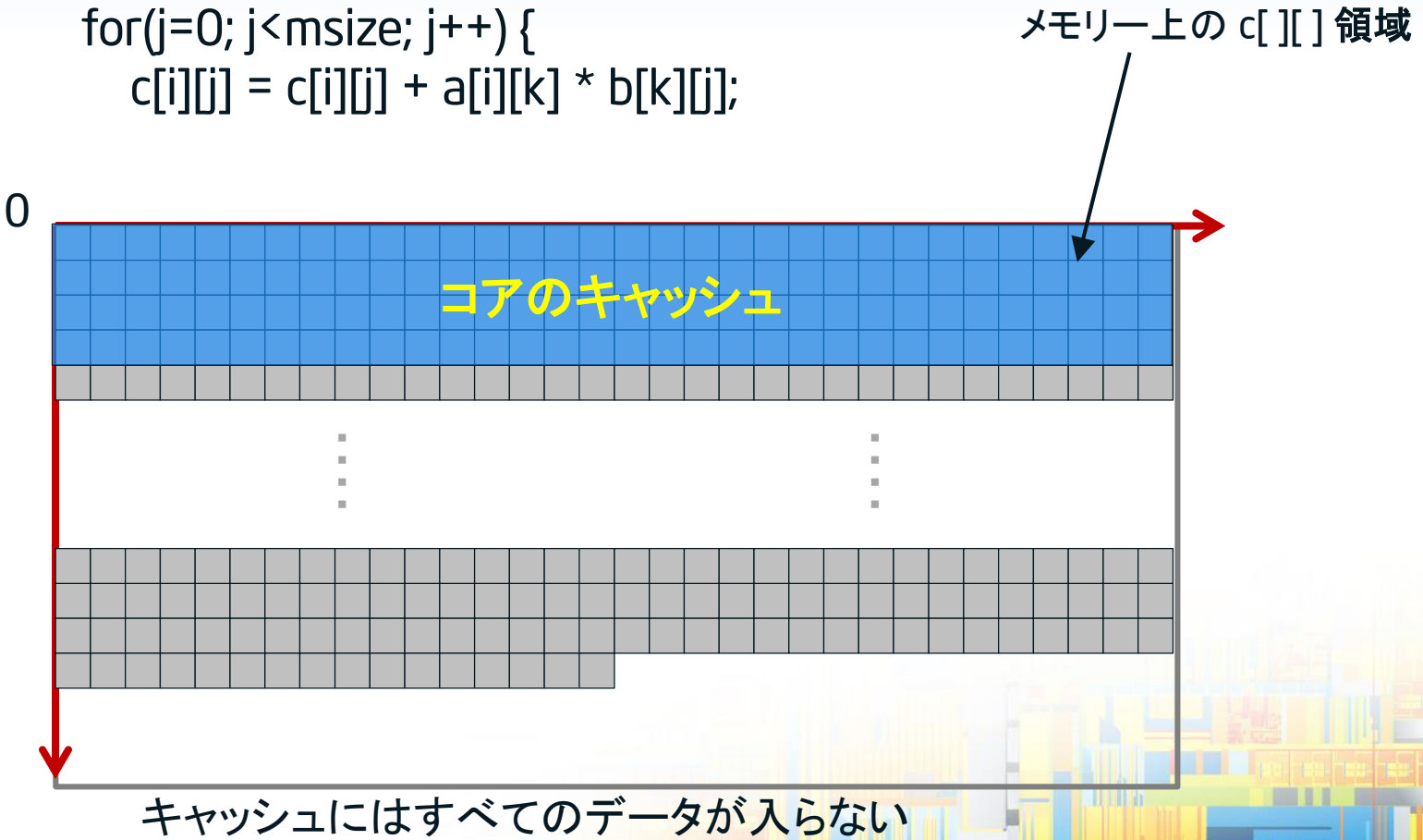
```
for (i=0; i<N; i++)
        A[j][i]
```

Code the Future

OPTIMIZATION NOTICE

メモリーとキャッシュサイズ

```
for(i=tidx; i<msize; i=i+numt) {
    for(k=0; k<msize; k++) {
        for(j=0; j<msize; j++) {
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

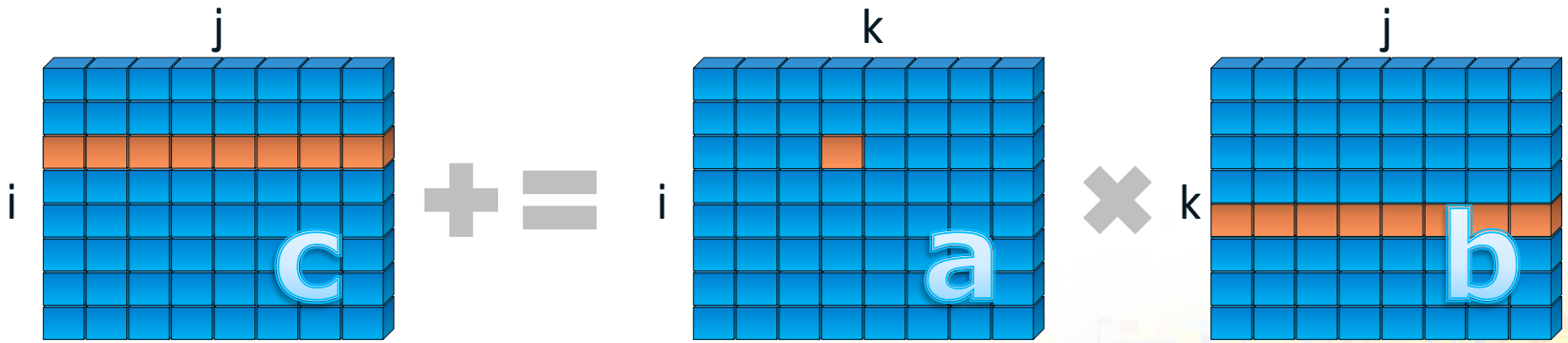


Code the Future

使用されるメモリーの状況

```
for(i=tidx; i<msize; i=i+numt) {
    for(k=0; k<msize; k++) {
        for(j=0; j<msize; j++) {
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

最内のループでは、以下のようにメモリーが使用されている



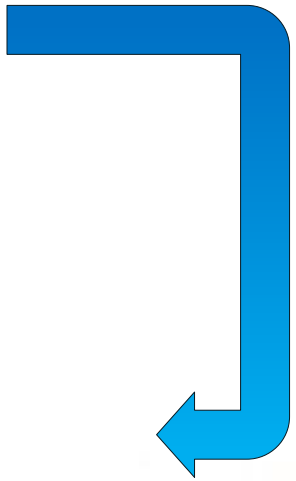
※メモリー上に展開される配列を簡略的に表した図

このような記述であれば、いずれキャッシュ容量を使い切ってしまう

キャッシュに着目した最適化

キャッシュ中の小さなブロックのデータにも影響するようにネストを再配置する

```
for(i=tidx; i<msize; i=i+numt) {
   for(k=0; k<msize; k++) {
      for(j=0; j<msize; j++) {
         c[i][j] = c[i][j] + a[i][k] * b[k][j];
```



ブロッキング

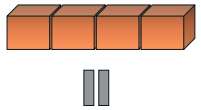
```
for (i0 = ibeg; i0 < ibound; i0 +=mblock) {
   for (k0 = 0; k0 < msize; k0 += mblock) {
      for (j0 =0; j0 < msize; j0 += mblock) {
         for (i = i0; i < i0 + mblock; i++) {
            for (k = k0; k < k0 + mblock; k++) {
               for (j = j0; j < j0 + mblock; j++) {
                  c[i][j]  = c[i][j] + a[i][k] * b[k][j];
```

Code the Future

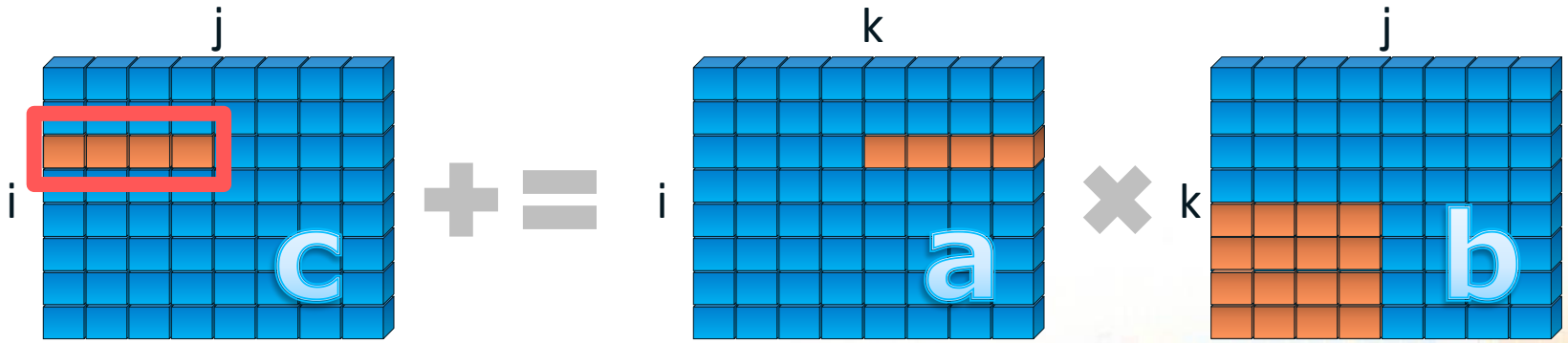
ブロッキングにより書き換えを防ぐ

```
for (i0 = ibeg; i0 < ibound; i0 +=mblock) {
   for (k0 = 0; k0 < msize; k0 += mblock) {
      for (j0 =0; j0 < msize; j0 += mblock) {
         for (i = i0; i < i0 + mblock; i++) {
            for (k = k0; k < k0 + mblock; k++) {
               for (j = j0; j < j0 + mblock; j++) {
                  c[i][j]  = c[i][j] + a[i][k] * b[k][j];
```

for ループの繰り返し回数を
msize から mblock に変更し、
アクセス範囲を限定する



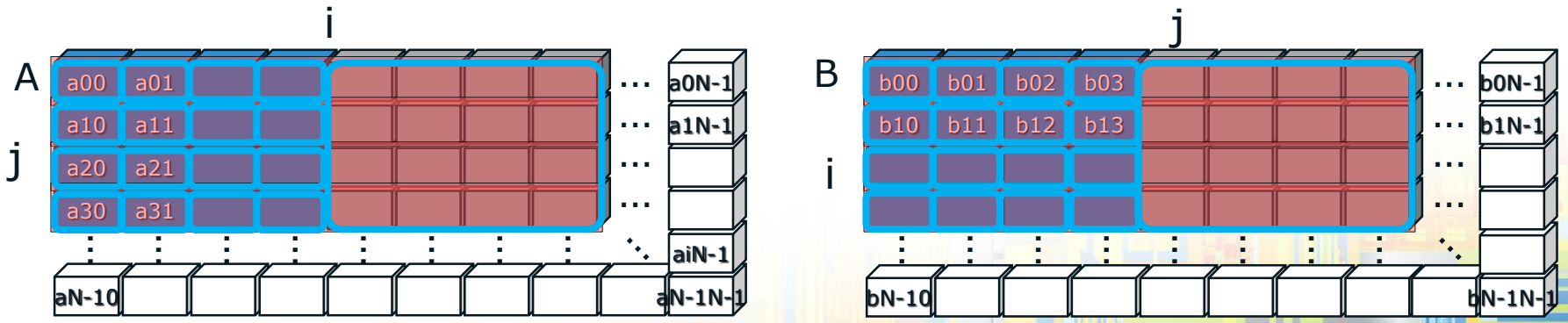
データタイプ × mblock 分
ただし、キャッシュラインをまたがない



キャッシュ上のデータを書き換えないようして (ブロッキング)
メモリーの読み込みとキャッシュ入れ替えによる遅延の発生を防ぐ。

他のブロッキング例

```
for(i=0;i<MAXI;i++)
  for(j=0;j<MAXJ;j++)
    A[j][i] = A[j][i] + B[i][j];

for(ii=0;ii<MAXI/BLOCKI;ii++)
    for(jj=0;jj<MAXJ/BLOCKJ;jj++)
      for(i=(ii*BLOCKI);i<((ii+i)*BLOCKI);i++)
        for(j=(jj*BLOCKJ);j<((jj+j)*BLOCKJ);j++)
          A[j][i] = A[j][i] + B[i][j];
```



読み込んだ A のキャッシュラインのデータが消えないように使用する

キャッシュに関するパフォーマンス問題の修正

Hotspots で使用されるデータについて以下の修正を検討する

・データ・アライメント

・キャッシュ・ブロッキング

・連想性 (set associativity) による L1 の 4KB のアクセス間隔または
　データサイズ、L2 の 64KB のアクセス間隔などは遅延を招くため、
　パディングバイトを追加する

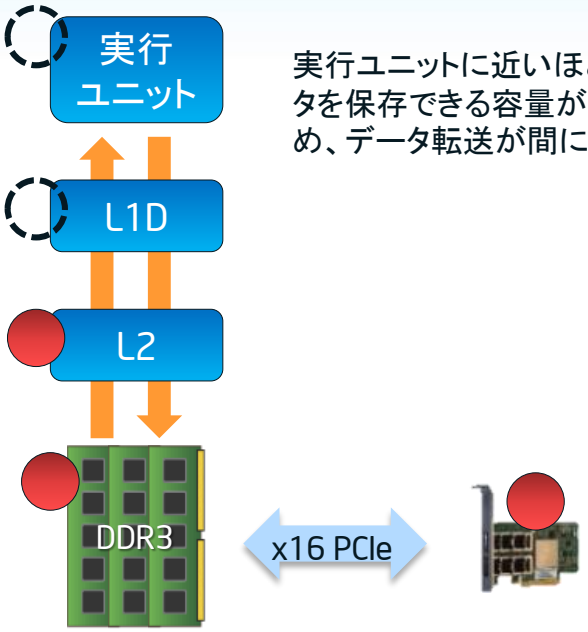
パディングバイトの追加
1 キャッシュライン分 (64byte) の余分なデータ要素を追加することで、キャッシュが頻繁に上書きされることによる遅延を回避することができる

Code the Future

さまざまなボトルネックの原因を特定

intel Software

キャッシュに関係するボトルネック



実行ユニットに近いほど、データを保存できる容量が小さいため、データ転送が間に合わない

演算に必要なデータがどこまで来ているかを確認し、コードを修正することによってパフォーマンスを改善させる

分岐予測ミスによるボトルネック

条件分岐などの予測ミスが生じると、プロセッサーのパイプラインに準備された命令を廃棄することになり、処理遅延が発生する。

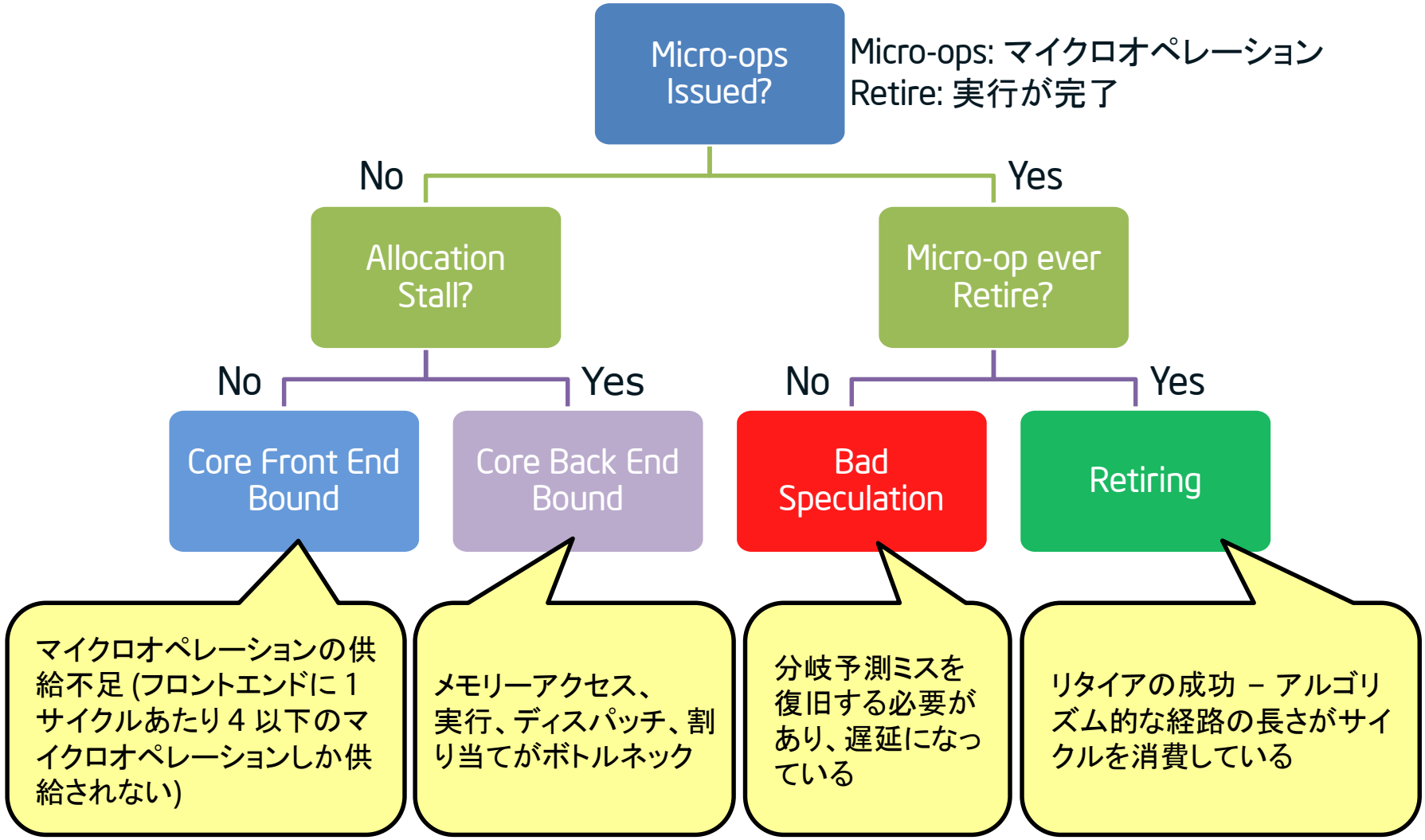
多発する分岐予測ミスがある箇所の分岐方法や処理方法を変更して、パフォーマンスを改善させる。

メモリーバンド幅に関するボトルネック

転送速度を確認し、規定を下回ればコードを修正する

その他、さまざまなボトルネックに対応

何に注目して最適化すべきか



- 解析タイプ “General Exploration” でのサンプリング結果を “General Exploration” ビューポイントで表示した場合の項目

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

Code the Future

OPTIMIZATION NOTICE