この資料の対象：ソフトウェア開発者

- アプリケーションのパフォーマンス最適化に注目する開発者
  - パフォーマンスのエキスパートである必要はなりません
  - しかし、アプリケーション自体のことは熟知していることが前提です
- 第 4 世代インテル® Core™ プロセッサーを搭載するプラットフォームを使用している
- インテル® VTune™ Amplifier XE パフォーマンス・アナライザーを使用する
  - ここで紹介するパフォーマンスに関する情報は、他のツール（PTU など）にも適用できますが、ここではインテル® VTune™ Amplifier XE に注目します

このスライドの使い方

- 一度スライドを読み通して、データ収集時に再度参照してください
- パフォーマンス解析は、何度かの繰り返しで達成されることを忘れないでください
- ソフトウェアの最適化は、以下を行ってから始めてください：
  - 任意のコンパイラー最適化オプションを適用（/O2、/QxAVX など）
  - 適切なワークロードを選択
  - 基準となるパフォーマンスの測定

第 4 世代インテル® Core™ プロセッサー・ファミリー向けのソフトウェア最適化にインテル® VTune™ Amplifier XE を使用する

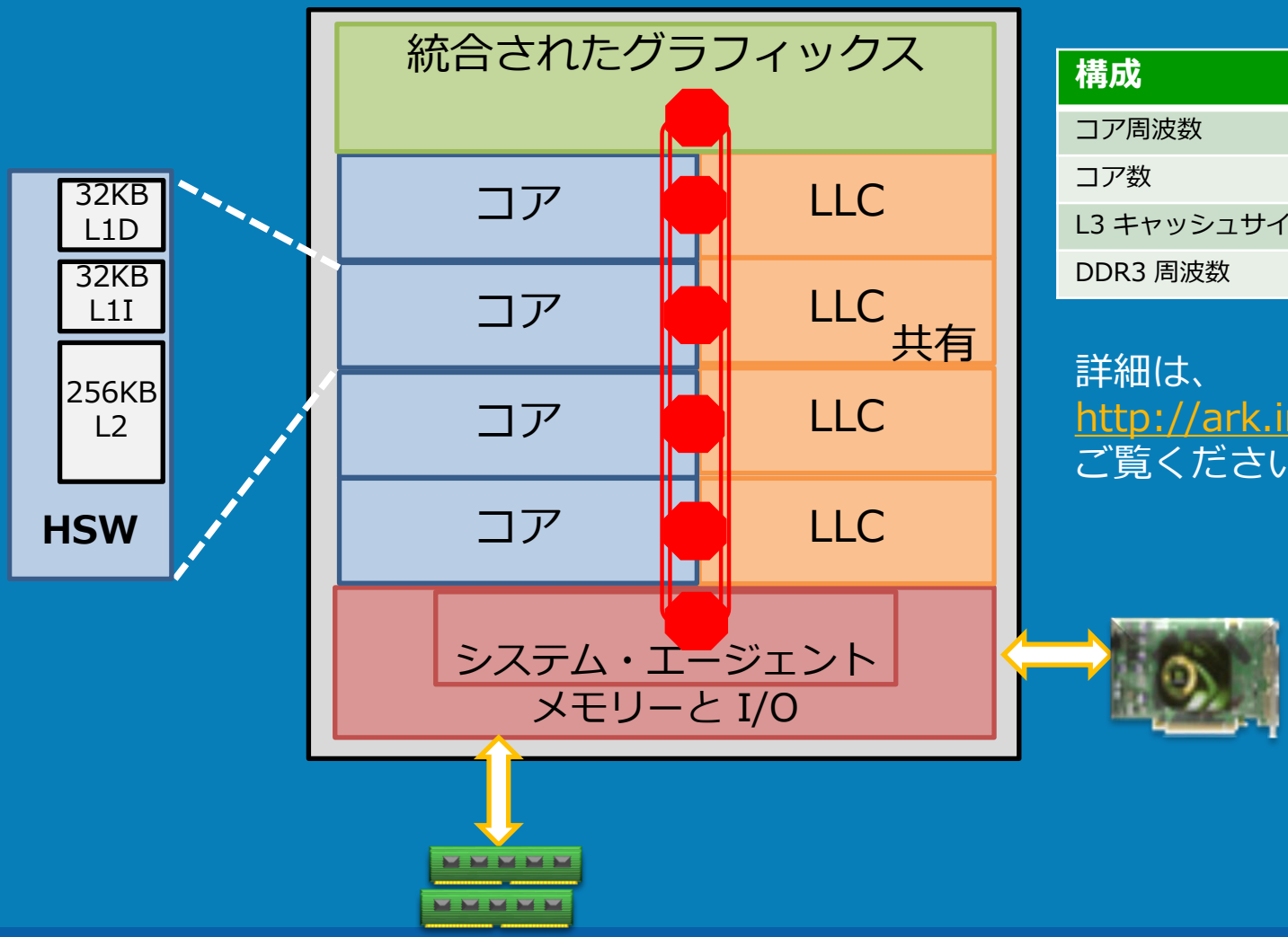
ソフトウェア & サービスグループ

Ver.1.0-J

内容

- 第 4 世代インテル® Core™ プロセッサー
- インテル® VTune™ Amplifier XE の特徴
- 新しいインテル® VTune™ Amplifier XE 2015
- ソフトウェア最適化サイクル
  - ホットスポット
  - 効率を判定する方法
- チューニングの薦め
  - フロントエンドのチューニング
  - バックエンドのチューニング
  - 投機のチューニング
  - リタイアのチューニング

第 4 世代インテル® Core™ プロセッサー



構成	
コア周波数	～ 3.4 GHz
コア数	～ 4 個
L3 キャッシュサイズ	～ 8M
DDR3 周波数	～ 1600 MHz

詳細は、
http://ark.intel.com/を
ご覧ください。

第 4 世代インテル® Core™プロセッサー・ファミリーのパフォーマンス機能

- 22nm プロセス技術で製造 - パフォーマンスと電力効率を改善
- インテル® ターボ・ブースト・テクノロジー 2.0
- インテル® ハイパースレッディング・テクノロジー
- インテル® アドバンスト・ベクトル拡張（インテル® AVX）およびインテル® アドバンスト・ベクトル拡張 2（インテル® AVX2）
- インテル® スマート・キャッシュ
- インテル® クイック・シンク・ビデオ
- インテル® クリアー・ビデオ HD
- インテル® InTru™ 3D

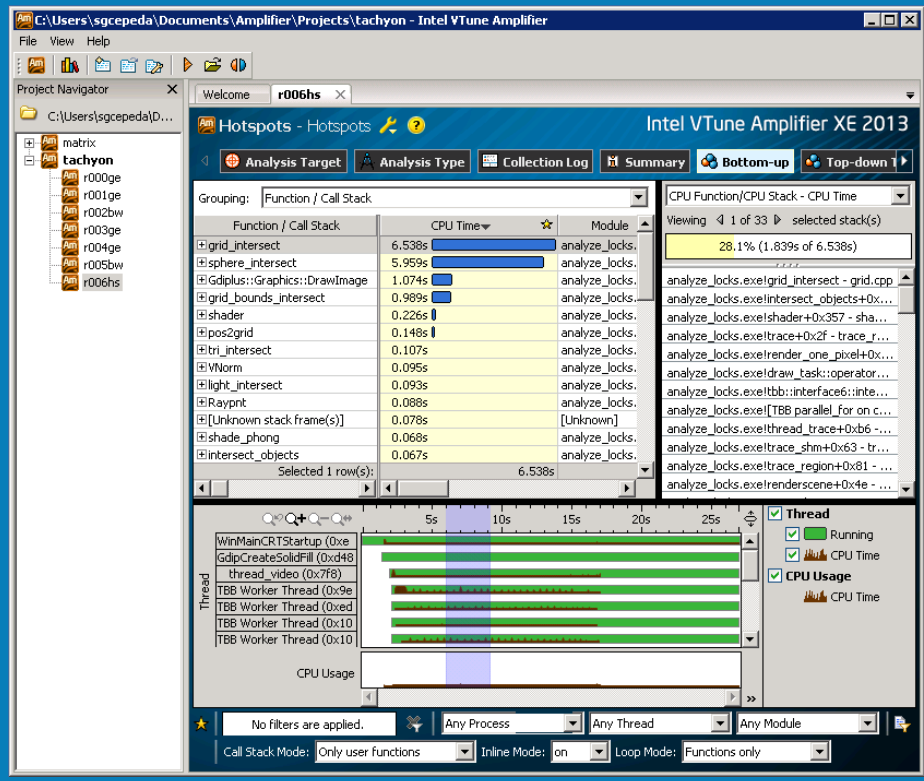
Haswell† マイクロアーキテクチャーを採用

†:コード名

インテル® VTune™ Amplifier XE の特徴

VTune Amplifier XE 2015 の機能：

- 複数の収集タイプ
  - >ホットスポット (統計呼び出しツリー)
  - >スレッドの並行性
  - >ロックと待機の分析
  - >イベントベース・サンプリング
- 全ての解析タイプを
  タイムライン表示に統合
- ソースとアセンブリー表示
- C/C++、Fortran、Java、アセンブリー、C# との互換性
- Visual Studio* への統合、コマンドラインでの利用、もしくは Windows* と Linux* 向けスタンドアロン・インターフェイスを利用

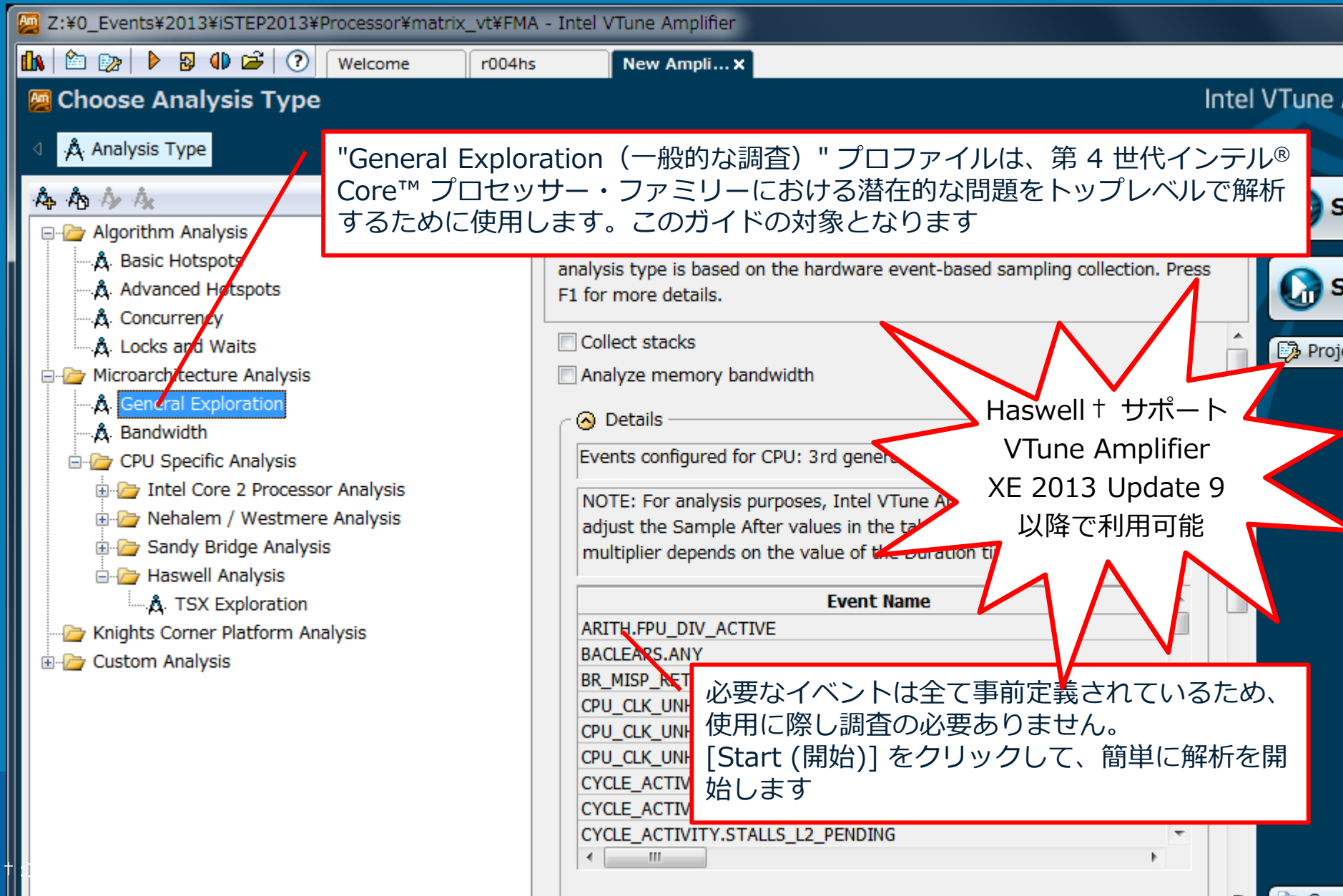


新しいインテル® VTune™ Amplifier XE

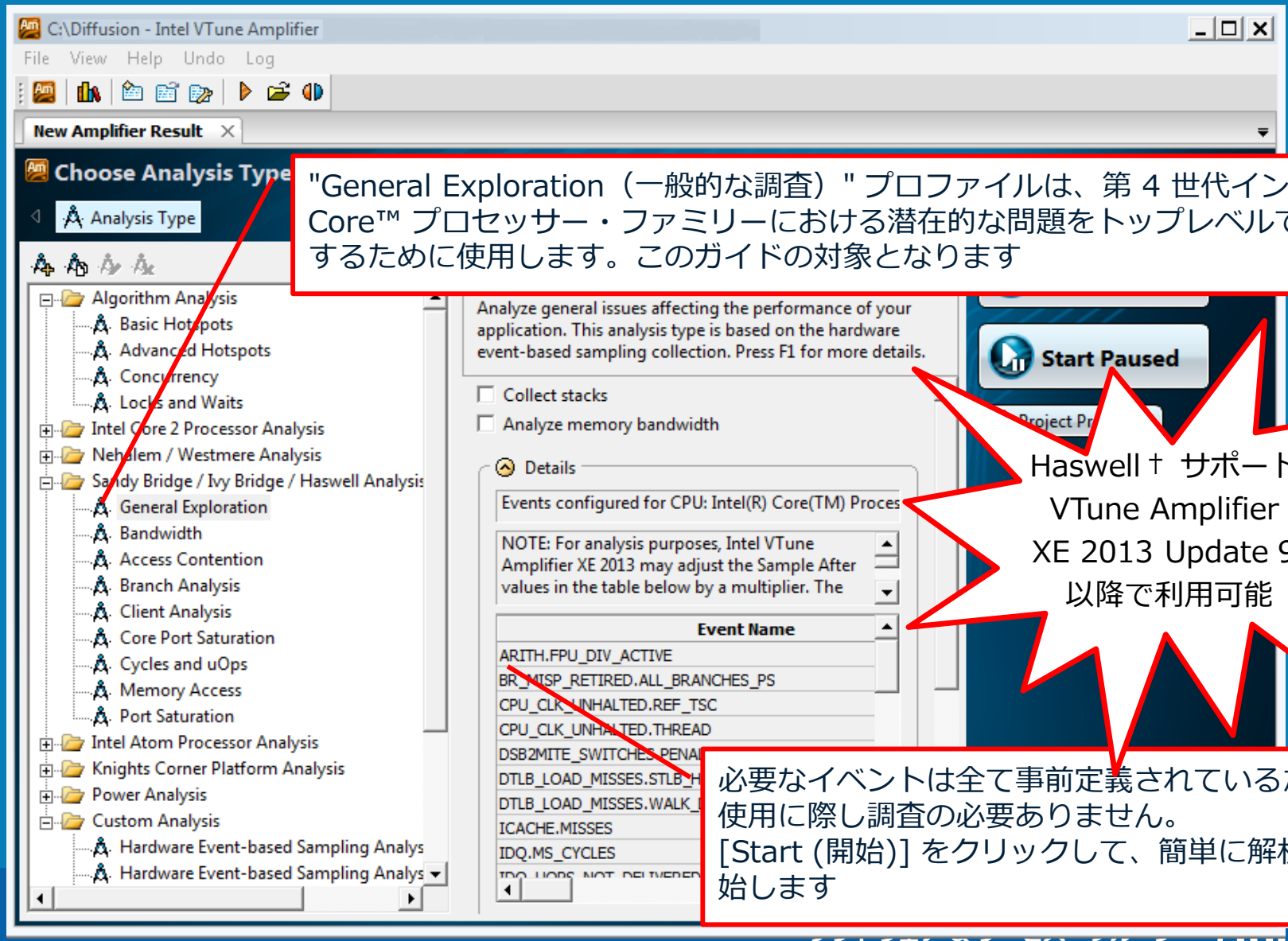
VTune Amplifier XE 2015 の新機能：

- OpenCL カーネルの解析をサポート
  - > SIMD Width プロファイルの追加
- OpenMP 解析の拡張
  - > 並列化で性能向上の可能性を CPU 利用率のヒストグラムと測定基準で提供
- インテル® Xeon Phi™ コプロセッサーの解析
  - > スタック情報を収集
  - > 解析ワークフローを改善
- カスタム・グループ・レベルの作成と表示
- OS X* 上で結果を解析（データ収集はできません）

Amplifier XE の新機能：定義済みプロファイル！



Amplifier XE 2013 での表示例



"General Exploration（一般的な調査）" プロファイルは、第 4 世代インテル® Core™ プロセッサー・ファミリーにおける潜在的な問題をトップレベルで解析するために使用します。このガイドの対象となります

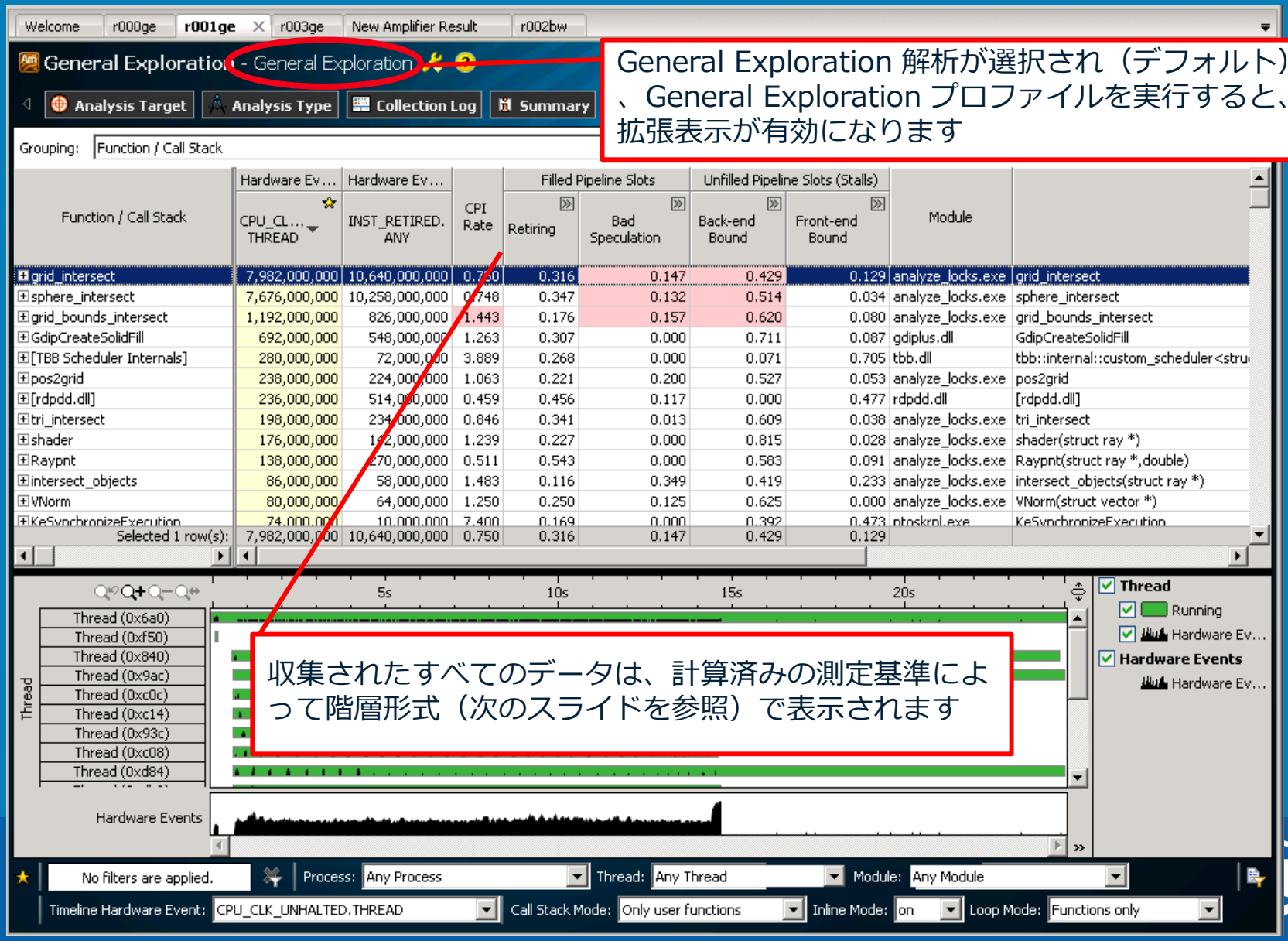
Haswell † サポート
VTune Amplifier
XE 2013 Update 9
以降で利用可能

必要なイベントは全て事前定義されているため、使用に際し調査の必要ありません。
[Start (開始)] をクリックして、簡単に解析を開始します

†:コード名

Haswell † マイクロアーキテクチャー向けの General Exploration 表示の拡張

†:コード名

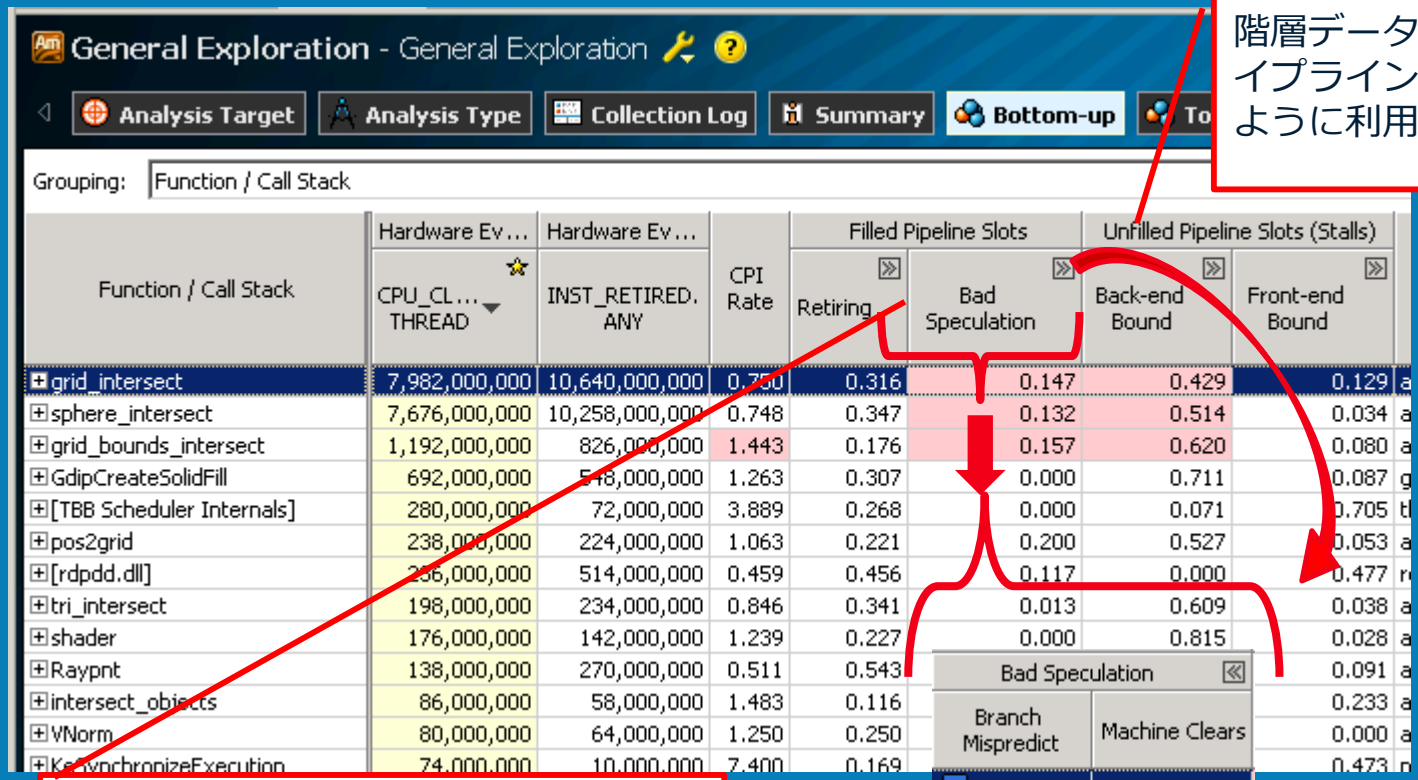


General Exploration 解析が選択され（デフォルト）、General Exploration プロファイルを実行すると、拡張表示が有効になります

収集されたすべてのデータは、計算済みの測定基準によって階層形式（次のスライドを参照）で表示されます

Haswell † マイクロアーキテクチャー向けの General Exploration 表示の拡張

†:コード名

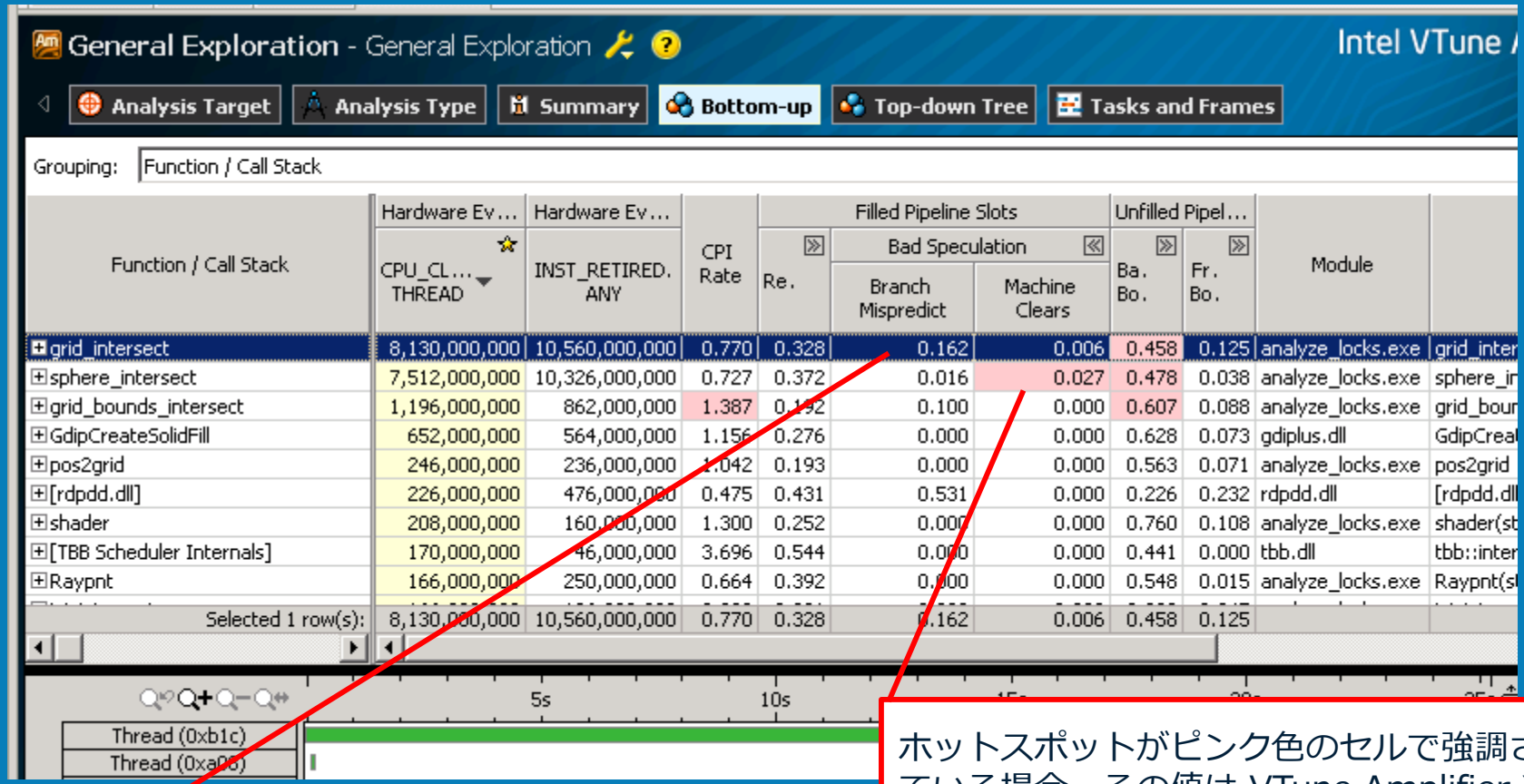


階層データの表示は、各コアのパイプラインで実行スロットがどのように利用されているか示します

パイプラインの利用率のカテゴリーに関連する内訳を見るため、カラムを拡張します：リタイアメント、投機の問題、バックエンドもしくはフロントエンドのパイプライン・スロット

Haswell † マイクロアーキテクチャー向けの General Exploration 表示の拡張

†:コード名



パイプライン利用率の各カテゴリーに表示される事前計算された測定基準は、利用者の解析時間を軽減します

ホットスポットがピンク色のセルで強調されている場合、その値は VTune Amplifier XE が持つしきい値を超えており、調査の必要があることを意味します

パフォーマンス計測の複雑性

- 第 4 世代インテル® Core™ プロセッサー・ファミリーには、パフォーマンス計測に大きな影響を与える 2 つの機能があります：
  - インテル® ハイパースレッディング・テクノロジー
  - インテル® ターボ・ブースト・テクノロジー 2.0
- これらの機能が有効にされている場合、パフォーマンス・データを計測して解釈するのが難しくなります
  - 大部分のイベントはスレッドごとにカウントされますが、いくつかのイベントはコアごとにカウントされます
    - 特定のイベントについては、インテル® VTune™ Amplifier XE のヘルプをご覧ください
- 一部の専門家は、これらの機能を無効にしてパフォーマンスを解析し、最適化が終了した後に再度有効にする手法を取ります

"ハードウェア上のソフトウェア" のチューニング手順

各ホットスポットで

- 効率を判断
  - > もし十分でなければ：
    - 主なボトルネックを特定
    - 非効率であるアーキテクチャー上の理由を見つけます
    - 問題を最適化します

繰り返します!

"ハードウェア上のソフトウェア" のチューニング手順

各ホットスポットで

- 効率を判断
  - > もし十分でなければ：
    - 主なボトルネックを特定
    - 非効率であるアーキテクチャー上の理由を見つけます
    - 問題を最適化します

繰り返します!

ホットスポットの特定

- **ホットスポットとは：**アプリケーションが多くの時間を費やしているコード中の場所
- **なぜ重要なのか：**そこに最適化の労力を費やすべきです！
  - >アプリケーションの実行時間の 2% にしか満たない関数を改善する必要がありますか？
- **どのように検出するか：**VTune Amplifier XE の、“Basic Hotspots” か “Advanced Hotspots” 解析タイプを利用します
  - >通常ホットスポットは、CPU_UNHALTED.THREAD イベント（別名 "clockticks"）で定義されます

"ハードウェア上のソフトウェア" のチューニング手順

各ホットスポットで

- 効率を判断
  - > もし十分でなければ：
    - 主なボトルネックを特定
    - 非効率であるアーキテクチャー上の理由を見つけます
    - 問題を最適化します

繰り返します!

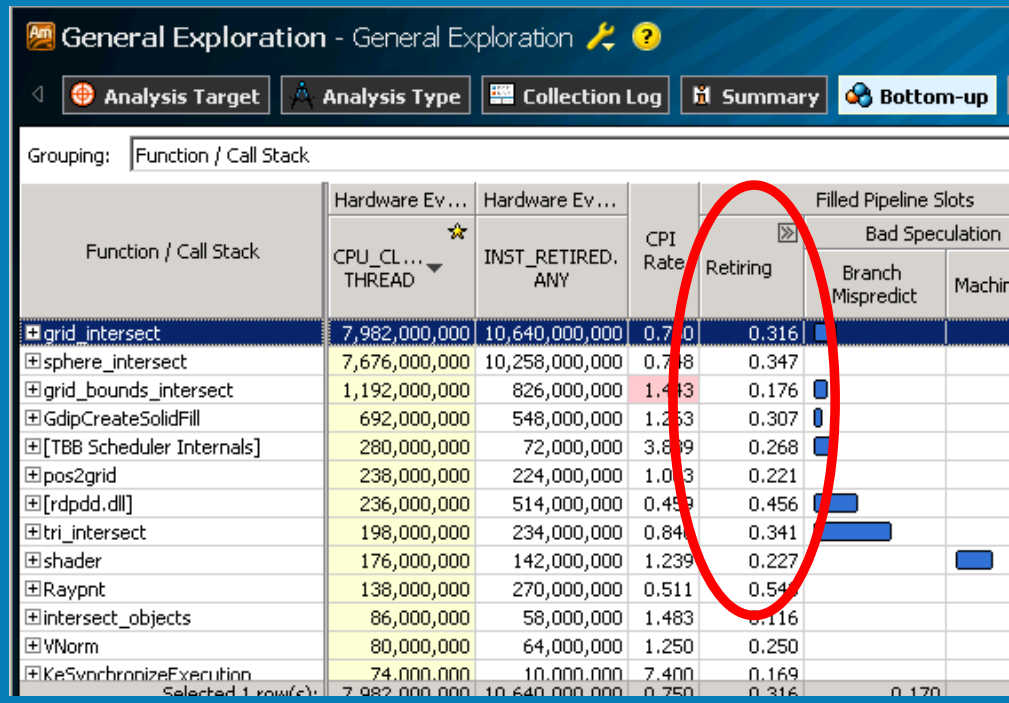
効率を判断

- 次の 3 つの方法のいずれかを使用して、ホットスポットの効率を判断：
  - リタイアしたパイプライン・スロットの比率 (%)
  - CPI の変化
  - コードを調査

- 注意：これらの手法の幾つかは、特定のコードにおいて他の方法よりも適しています。以降のスライドを参照して下さい

効率法 1:
リタイアしたパイプライン・スロットの比率（%）

- **なぜ重要なのか：**アプリケーションがプロセッサーを使用する効率を理解するのに役立ちます
- **どのように検出するか：**General Exploration プロファイルの測定基準 : "*Retiring*"（リタイアした比率）に注目
- **何を行うか：**
  - 特定のホットスポットの場合：
  - 75% 以上のパイプライン・スロットがリタイアした場合（.75 以上）、3 番目の効率法に進み、ベクトル化によってパフォーマンスを向上できるか確認するには、コードスタディー 1 を参照します
  - それ以外は、次のスライドに進みます



効率法 1:
リタイアしたパイプライン・スロットの比率（%）

- **何を行うか：**パイプライン・スロットのリタイア比率が 75% 以下のホットスポットは、効率を判断する際にアプリケーションのタイプを考慮してください。ホットスポットが期待される範囲以下である場合、それは非効率であると言えます

	各カテゴリーのパイプライン・スロットの期待される範囲（うまく最適化されたホットスポット向け）：		
カテゴリー	クライアントおよびデスクトップ向けアプリケーション	サーバー、データベースおよび分散型アプリケーション	ハイパフォーマンス・コンピューティング（HPC）アプリケーション
Retiring (リタイアの比率)	**20-50%**	**10-30%**	**30-70%**

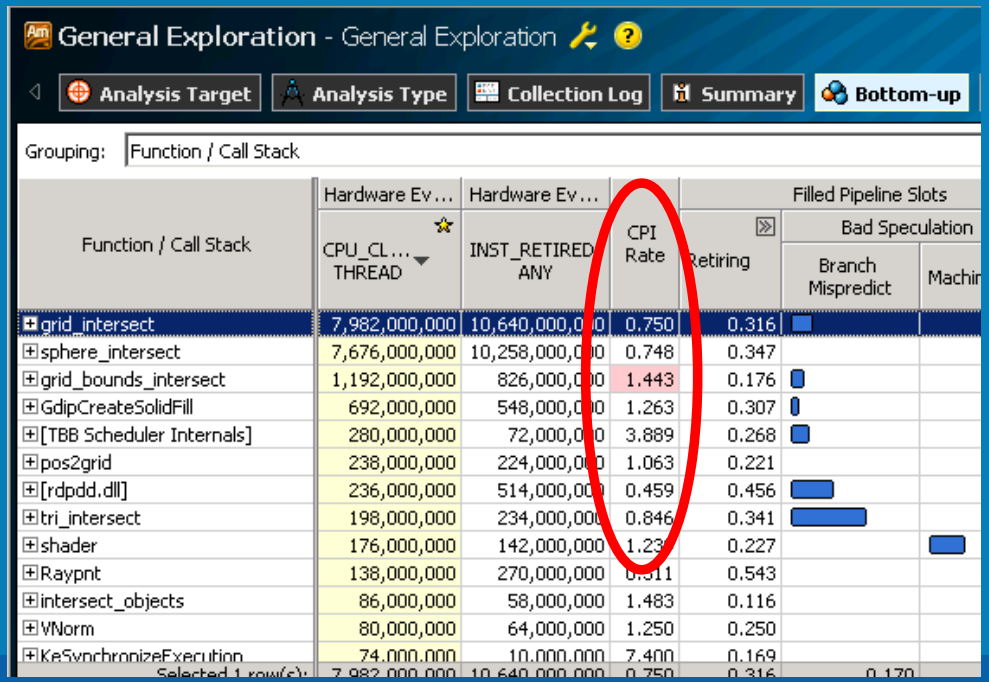
詳しい情報は以下をご覧ください：http://software.intel.com/en-us/articles/how-to-tune-applications-using-a-top-down-characterization-of-microarchitectural-issues

効率法 2:
命令あたりのサイクル数（CPI）の変化

- **なぜ重要なのか：**2 つのデータセットを比較する際に役立つ、もう一つの効率の尺度

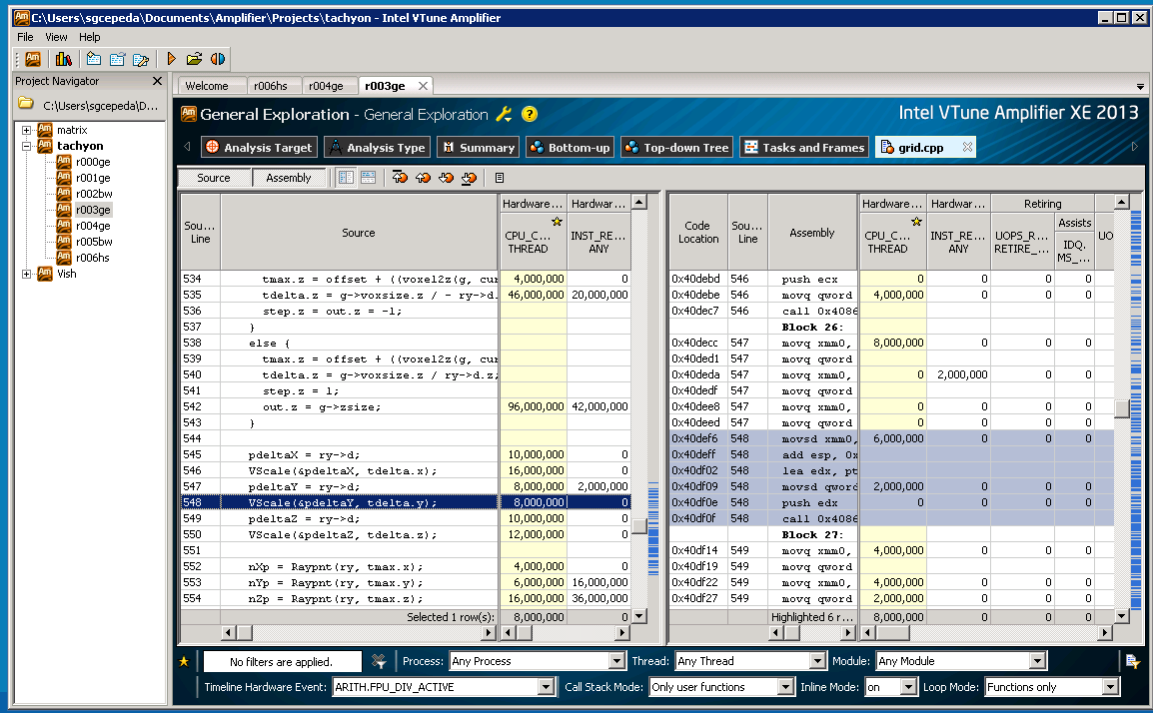
  >アプリケーションのワークロードの 一部を実行する命令の平均クロックを表示します

- **どのように検出するか：**General Exploration プロファイルの測定基準："*CPI*" を調査します
- **何を行うか：**
  - CPI は、アプリケーションとプラットフォームに依存し大きく変化します
  - コードサイズが変わらないなら、最適化では CPI を減らすことに注目すべきです



効率法 3:コードを調査

- **なぜ重要なのか：**効率法の 1 と 2 では、実行される命令の時間を測定しました。非効率である他の原因は、非常に多くの命令を実行することです
- **どのように検出するか：**VTune Amplifier XE のソースと逆アセンブリー表示を利用します
- **何を行うか：**
  - コードサイズが大きく、新しい命令セットが利用できていないことが考えられます
  - 問題の可能性を次の 3 枚のスライドで説明します



効率法 3、コードスタディー 1：旧式の浮動小数点もしくは整数コードをインテル® アドバンスト・ベクトル拡張（AVX と AVX2）に変換します

- **なぜ重要なのか：**SIMD 命令を使用することで、浮動小数点演算のパフォーマンスを大幅に向上できます。既存の FP もしくは整数 SSE コードを AVX 命令に変換すると、広いベクトルデータ（256 ビットまで）のサポート、NDS 操作を可能にする 3 もしくは 4 オペランドの構文、そして電力節約などいくつかの利点があります
- **どのように検出するか：**既存の SSE 命令（xmm レジスターを検索）、MMX 命令（mm レジスターを検索）、もしくは x87 浮動小数点命令のアセンブリー・コードを調査します。パック操作を行っていない（faddp、fmul や addss などのスカラー SSE 命令）コードを特定します
- **何を行うか：**

  –インテル® コンパイラー：/QxCORE-AVX2 (Windows*) や -xAVX2 (Linux*) オプションを使用

  –GCC の -march=core-avx2 オプションを使用

  –AVX への最適化に関しては、 Intel® 64 and IA-32 Architectures Optimization Reference Manual の、第 11 章をご覧ください

効率法 3、コードスタディー 2：
Fused Multiply Add (FMA) 命令の恩恵を得る

- **なぜ重要なのか：** Haswell† の FMA 命令は、FP 乗算命令とレイテンシーは同じです。新たな 2 つの FMA ユニットは、2倍のピーク FLOPs/サイクルを可能にします
- **どのように検出するか：** r=(x*y)+z、r=(x*y)-z、r=-(x*y)+z、もしくは r=-(x*y)-z 形式の操作を行うソースコードを探します。該当するアセンブリーコードを見て、FMA 命令が使用されているか調べます。FMA 命令は、**VFM**（VFMADD132PD や VFMSUBADD132PD のように）もしくは **VFNM** (VFNMADD132PD のように）という名前で始まります
- **何を行うか：**

  –FMA を生成するため、コンパイラーのオプションを使用します（精度のゆるい浮動小数点モデルを利用した方がいいでしょう）

  –インテル® コンパイラー：-fma もしくは /Qfma と CORE-AVX2 以上の引数を持つ -x か /Qx 、または -march (Linux*) か /arch (Windows*) オプション

  –GCC：-xfma もしくは -march=core-avx2 オプション

† :コード名

効率法 3、コードスタディー 3：インテル® AES-NI (インテル® Advanced Encryption Standard New Instruction) の改善を活用

- **なぜ重要なのか：**インテル® AES-NI は、一部の機能をハードウェアで実装することで AES アルゴリズムの実行を高速化します
- **どのように検出するか：**アプリケーションが AES アルゴリズムを実装している場合、インテル® AES-NI が使用されていることを確認してください。オペランドとして xmm レジスターを持つ AES 命令のブロックは、並列モデルを使用している場合があります：

```
aesenc   %xmm5, %xmm7
aesenc   %xmm5, %xmm8
aesenc   %xmm5, %xmm9
aesenc   %xmm5, %xmm10
```

- **何を行うか：**
  - インテル® AES-NI が、並列モデル（ECB、CTR そして CBC-複合化など）で利用されている場合、並列で処理するブロックの命令数を再定義（Haswell† では 8 個）することでパフォーマンスを向上できます
  - アプリケーションが暗号化や複合化を行っていても、インテル® AES-NI が使用されていない場合、ぜひ試してみてください。 http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni/ を参照

†:コード名

"ハードウェア上のソフトウェア" のチューニング手順

- 各ホットスポットで
  - 効率を判断
    - > もし十分でなければ：
      - 主なボトルネックを特定
      - 非効率であるアーキテクチャー上の理由を見つけます
      - 問題を最適化します
- 繰り返します!

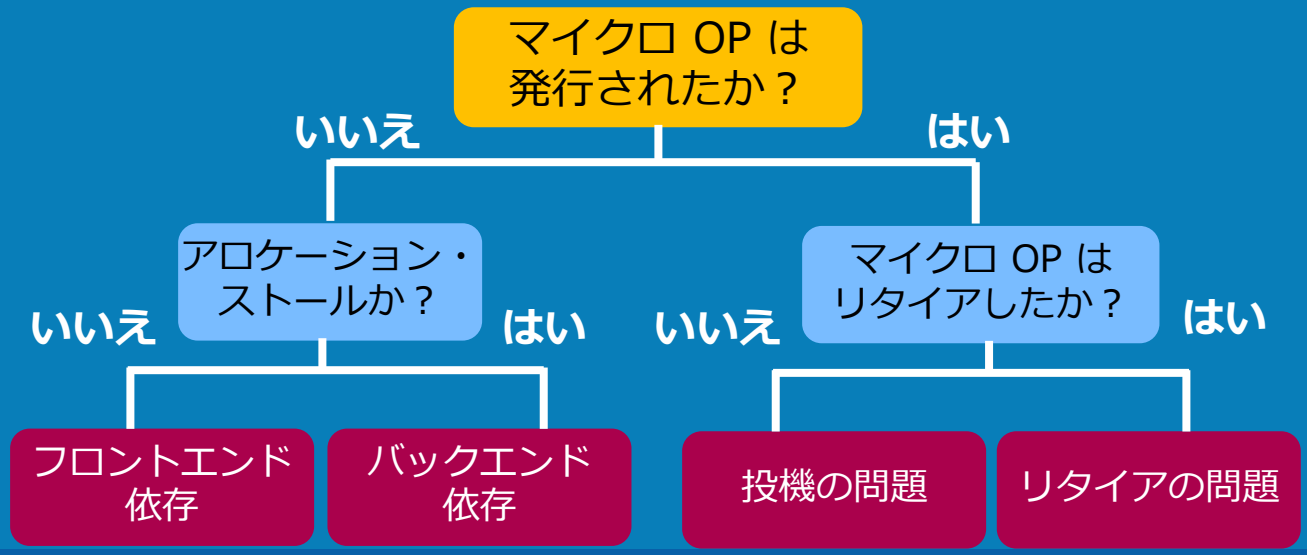
主なボトルネックを特定

- コードが非効率であるかどうかを調べるため、効率法 1 もしくは 2 を適用している場合、最初に主要なボトルネックを見つけます
- General Exploration で実装されるトップダウンの階層は、アプリケーションの利用率を CPU コアの 4 つのカテゴリーに分類します：
  - フロントエンド依存
  - バックエンド依存
  - 投機の問題
  - リタイア

主要なボトルネックは、パイプラインの断片化が高く、最初に調査しなければいけません

問題の分類

- パイプライン・スロットは、抽象的なコンセプトであり、1 つのマイクロ・オペレーションを操作するのに必要なハードウェアのリソースを表します
- Haswell† では、各コアごとに各サイクルで 4 つのパイプライン・スロットがあります
- アプリケーションやホットスポットで利用可能な各パイプライン・スロットで何が起こっているかに応じて、パフォーマンスは分類されます



†:コード名

ボトルネックの調査

- どのカテゴリーが主要なボトルネックであるかを判断し、次にそのカテゴリーを以下の表と比較します
  - コードのホットスポットの比率（%）が範囲外であれば、そのカテゴリーから問題を調査します

	各カテゴリーのパイプライン・スロットの期待される範囲（うまく最適化されたホットスポット向け）：		
カテゴリー	クライアントやデスクトップ向けアプリケーション	サーバー、データベースおよび分散型アプリケーション	ハイパフォーマンス・コンピューティング（HPC）アプリケーション
スライド 53 **Retiring (リタイアの比率)** ↑	**20-50%**	**10-30%**	**30-70%**
スライド 37 **バックエンド依存** ↓	**20-40%**	**20-60%**	**20-40%**
スライド 33 **フロントエンド依存** ↓	**5-10%**	**10-25%**	**5-10%**
スライド 49 **投機の問題** ↓	**5-10%**	**5-10%**	**1-5%**

要点 ↑ **高い値が良い**
↓ **低い値が良い**

"ハードウェア上のソフトウェア" のチューニング手順

各ホットスポットで

- 効率を判断
  - > もし十分でなければ：
    - 主なボトルネックを特定
    - 非効率であるアーキテクチャー上の理由を見つけます
    - 問題を最適化します

繰り返します!

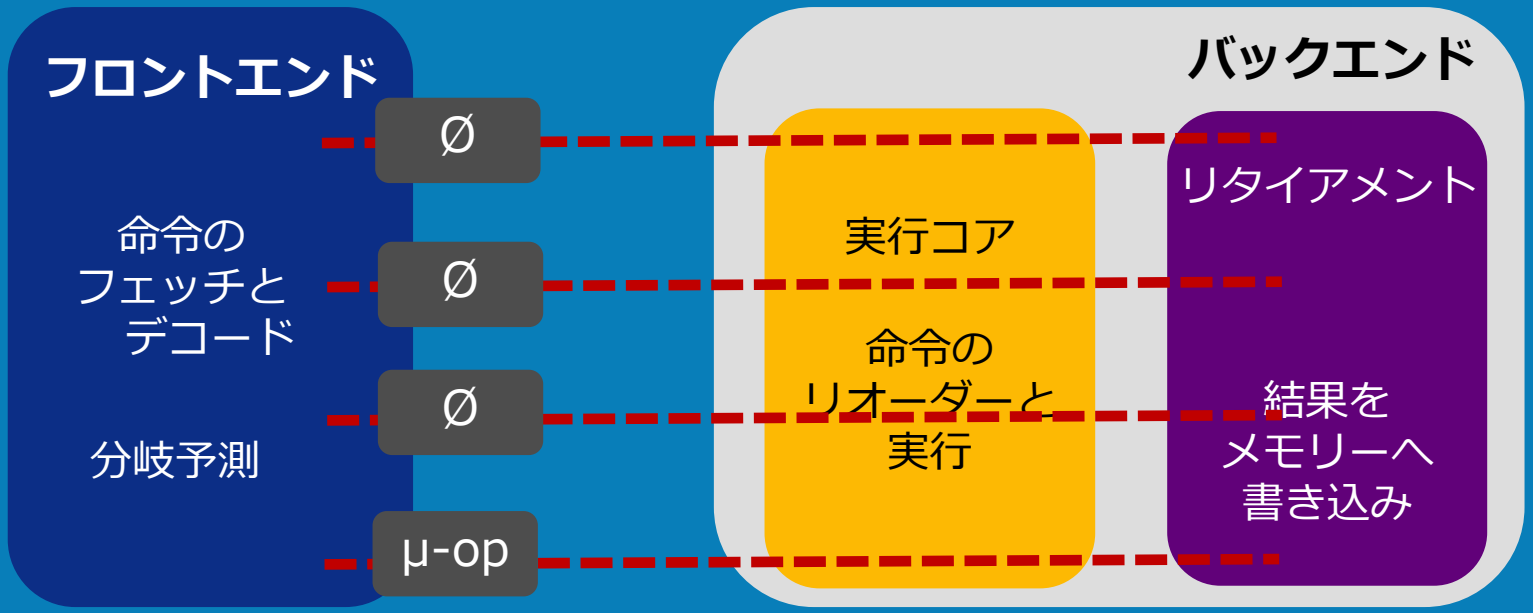
非効率であるアーキテクチャー上の理由を見つけます：以降のチューニング・スライド

- 問題はカテゴリー別にリストされ、カテゴリーごとに説明されます
- 潜在的なそれぞれの問題ごとに、いくつかの重要な情報が示されます
  - なぜ？ – なぜ潜在的な問題について考えなければいけないのか
  - どのように？– VTune Amplifire XE のどのプロファイルと測定基準を使用するか。表示項目がハイライトされていると、調査する必要があります
  - 何を行うか？– **問題を最適化する**のに役立ちます。調査を続行、もしくは最適化すべき提案を提示します
  - イベント名と測定基準の式は、別途記載します。VTune Amplifier XE のロジックに組み込まれているため、このスライドには含まれません。開発者は、事前定義されたプロファイルと測定基準のみを使用して、問題があるかどうかを指摘されるだけです

フロントエンド依存のカテゴリーをチューニング

パイプラインのフロントエンド
- 命令をフェッチ
- 命令をマイクロ・オペレーションにデコード
- サイクルごとに最大 4 マイクロ・オペレーションをバックエンドへ供給

フロントエンドの概要



フロントエンドが、4 つのパイプライン・スロット全てに
マイクロ・オペレーション(µ-op) を供給できない時に発生します。
一般に、コードのフェッチや命令デコードの遅延が原因です

インテル® VTune™ Amplifier XE でフロントエンド階層をブレークダウン

Filled Pipeline Slots		Unfilled Pipeline Slots (Stalls)	
Retiring	Bad Speculation	Back-end Bound	Front-end Bound
0.803	0.006	0.140	0.051
0.208	0.010	0.730	0.053
0.105	0.072	0.563	0.260
0.437	0.000	0.540	0.224
0.052	0.040	0.706	0.202
0.132	0.000	0.550	0.318

フロントエンド依存のカラムを展開して、マイクロ・オペレーションが供給されなかった "Front-End Latency" と 4 マイクロ・オペレーション以下が供給された "Front-End Bandwidth" に分類されるフロントエンド依存の比率を確認できます

Front-end Bound	
Front-End Latency	Front-End Bandwidth
0.006	0.045
0.029	0.024
0.183	0.076
0.207	0.017
0.231	0.000
0.144	0.074

フロントエンド依存が主要なボトルネックである場合、フロントエンド・レイテンシーに注目します。このカテゴリー下のハイライトされた問題を解決します

Front-end Bound				
Front-End Latency			Front-End Bandwidth	
ICache Misses	ITLB Overhead	Length Changing Prefixes	Front-End Bandwidth DSB	Front-End Bandwidth MITE
0.000	0.000	0.000	0.011	0.002
0.000	0.000	0.000	0.324	0.031
0.157	0.016	0.017	0.044	0.122
0.000	0.063	0.069	0.000	0.000
0.000	0.029	0.069	0.000	0.392
0.000	0.000	0.000	0.000	0.864

フロントエンド・レイテンシー

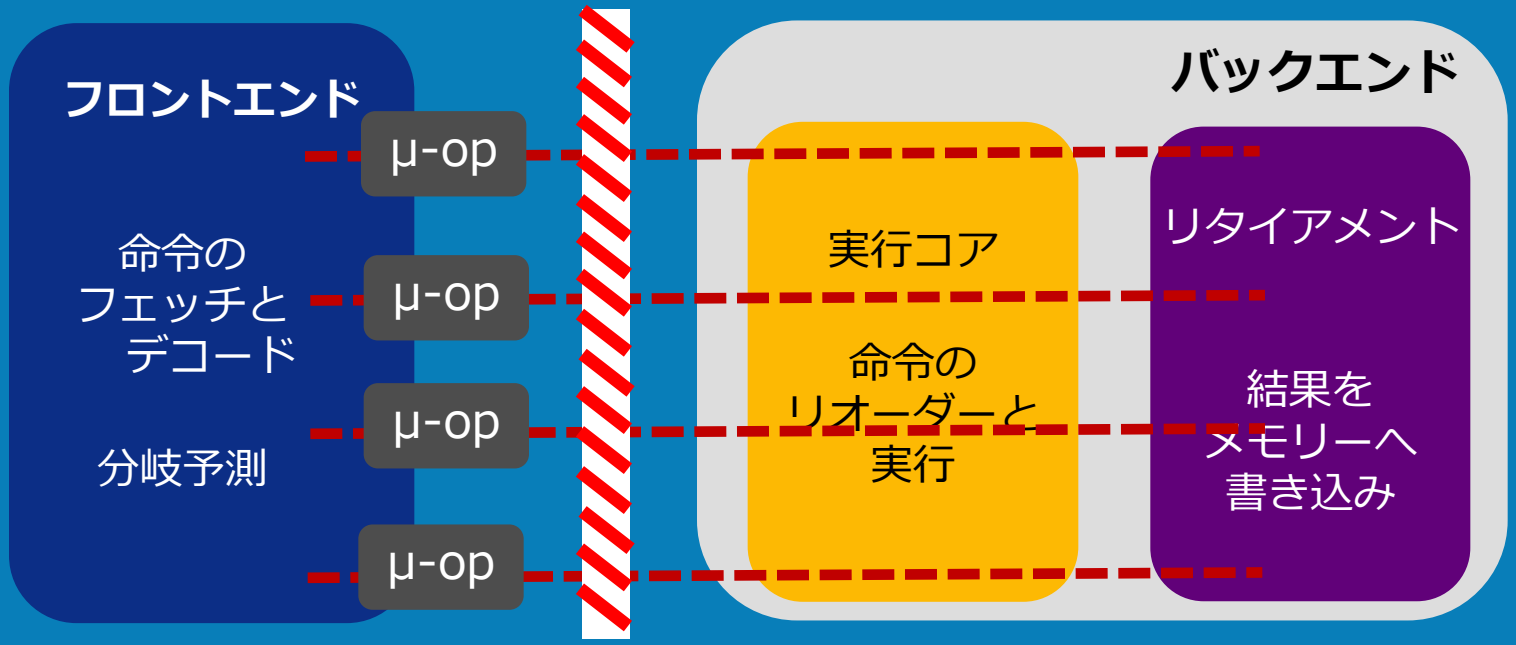
- **なぜ重要なのか：**フロントエンド・レイテンシーは、バックエンドで実行するマイクロ・オペレーションが不足する原因となります（命令スタベーション）
- **どのように検出するか：**フロントエンド・レイテンシーのサブカテゴリー、測定基準："*ITLB Overhead*"、"*ICache Misses*"、"*Length-Changing Prefixes*" を調査します
- **何を行うか：**
  - ホットスポットでこれらの測定基準がハイライトされている場合、より良いコード配置と生成テクニックを試みます：
    - コンパイラーのプロファイルに基づく最適化（PGO) を試してください
    - リンカーの順序付けテクニックを使用します（Microsoft* リンカーでは /ORDER、GCC ではリンカースクリプト）
    - コードサイズを減らすには、/O1 や /Os オプションを使用します
    - 動的に生成されたコードは、ホットなコードを近い場所に配置し、間接呼び出しを回避するようにしましょう

バックエンド依存のカテゴリーをチューニング

パイプラインのバックエンド

- フロントエンドからのマイクロ・オペレーションを受け取る
- 実行ユニット内の実行をスケジュールするため、必要に応じて命令をリオーダー（入れ替え）する
- メモリーから必要なオペランドを取り込む
- 操作を実行
- 結果をメモリーへ書き込む

フロントエンドの概要



**バックエンドが 4 つのパイプライン・スロット全てに
マイクロ・オペレーション(μ-op) を受け入れることができない時に発生。
通常、内部構造がすべて μ オペレーションで埋まっていてデータ待ちの状態**

インテル® VTune™ Amplifier XE におけるバックエンド階層のブレークダウン

Filled Pipeline Slots		Unfilled Pipeline Slots (Stalls)	
Retiring	Bad Speculation	Back-end Bound	Front-end Bound
0.803	0.006	0.140	0.051
0.208	0.010	0.730	0.053
0.105	0.072	0.563	0.260
0.437	0.000	0.540	0.224
0.052	0.040	0.706	0.202
0.132	0.000	0.550	0.318

Back-end Bound	
Memory Bound	Core Bound
1.000	0.000
1.000	0.000
0.943	0.000
0.874	0.000
1.000	0.000
0.768	0.000

バックエンド依存のカラムを展開して、多くのメモリー操作が実行中であるためバックエンドが新しい μ オペレーションを受け入れることができないことを示す “Memory Bound” と実行ポートの飽和問題を示す "Core Bound" に分類されるバックエンドの全ての比率を確認

いくつかの問題は、バックエンドのメモリー依存時間になることがあります。バックエンド時間の 50% 以上がメモリー依存に費やされている場合、ハイライトされた問題を調査します。50% 以上がコア依存に費やされている場合は、以降のスライドを参照してください

Back-end Bound													
Memory Bound										Core Bound			
Store Bo...	L1 Bound					L3 Bound				Port Utilization			
Split...	DTL...	DTL...	Loa...	Split...	4K ...	Con...	Data...	LLC ...	LLC ...	Cycl...	Cycl...	Cycl...	Cycl...
0.000	0.016	0.031	0.000	0.000	0.301	0.000	0.000	0.007	0.002	0.023	0.000	0.000	0.023
0.000	0.002	0.036	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.445	0.074	0.000	0.499
0.000	0.104	0.026	0.006	0.000	0.037	0.000	0.006	0.314	0.219	0.515	0.000	0.026	0.472
0.000	0.000	0.023	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.460	0.000	0.069	0.437
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.692	0.000	0.231	0.485
0.000	0.025	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.600	0.000	0.456	0.432

キャッシュミス

- **なぜ重要なのか：**キャッシュミスは、アプリケーションの CPI を高めます
  レベル 2 もしくは 3 ミスによる長いレイテンシーのデータに注目します
- **どのように検出するか：**
  メモリー依存のカテゴリー、測定基準：*”LLC Hit”*、*”LLC Miss”* を調査します
- **何を行うか：**
  - ホットスポットでどちらの測定基準もハイライトされている場合、ミスを減らすことを考慮する：
    - データ記憶域を減らすためアルゴリズムを変更
    - キャッシュに収まるようにブロック化してデータをアクセス
    - 共有の問題を調査（アクセス競合を参照）
    - ベクトルデータをアライメントし（そしてコンパイラーにそれを知らせます）
    - キャッシュライン入れ換え解析の概要に関しては、インテル® 64 および IA-32 アーキテクチャー最適化リファレンス・マニュアル の B.3.4.2 節をご覧ください
    - ストリーミング・ストアを利用
    - ソフトウェア・プリフェッチ命令を利用

アクセス競合

- **なぜ重要なのか：**コア間にまたがる（L2 レベルで）共有されるデータの変更は、データアクセスのレイテンシーを増加させます
- **どのように検出するか：**メモリー依存のサブカテゴリー、測定基準：*"Contested Accesses"* を調査
- **何を行うか：**
  - ホットスポットでこの測定基準がハイライトされた場合、ソース表示から HITM を生成しているソースコードの行を見つけます。HITM を生成した命令の次の命令にタグ付けされる "MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HITM_PS" イベントを探します
  - コードを詳しく調べて、真の共有か偽りの共有かを判断します
    適切な対策を施します：
    - 真の共有に対しては、共有の要求を減らします
    - 偽りの共有には、キャッシュライン境界にパディングを挿入します

データ共有

- **なぜ重要なのか：**（L2 レベルで）コア間のクリーンなデータ共有（読み込み共有）は、一貫性を維持するため少なくとも最初の参照でペナルティーが発生します
- **どのように検出するか：**
  メモリー依存のカテゴリー、測定基準： *"Data Sharing"* を調査します
- **何を行うか：**
  - ホットスポットでこの測定基準がハイライトされた場合、ソース表示から HIT を生成しているソースコードの行を見つけます。HIT を生成した命令の次の命令にタグ付けされる "MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HIT_PS" イベントを探します
  - コードを詳しく調べて、真の共有か、偽りの共有かを判断します
    適切な対策を施します：
    - 真の共有に対しては、共有の要求を減らします
    - 偽りの共有には、キャッシュライン境界にパディングを挿入します

その他のデータアクセスの問題：ストア・フォワードが行われないことによるロードのブロック

- **なぜ重要なのか：**ストアの結果がパイプラインを介してフォワードできない場合、依存するロードがブロックされることがあります
- **どのように検出するか：**メモリー依存のサブカテゴリー、測定基準：*"Loads Blocked by Store Forwarding"* を調査します
- **何を行うか：**
  - ホットスポットで測定基準がハイライトされている場合、ミスを減らすことを考慮します：
  - ソースを参照し、"LD_BLOCKS_STORE_FORWARD" イベントを探します。通常このイベントは、ブロックされたロードの次の命令にタグ付けされます。ロード命令の位置を確認し、フォワードできないストア命令を探します。通常 10 -15 命令以内にあります。もっとも一般的なケースは、同じアドレスに対するロードよりも小さな断片をストアする場合です。後続のロードと同じ、もしくは大きなサイズのデータをストアすることで問題を解決します

その他のデータアクセスの問題：キャッシュライン分割

- **なぜ重要なのか：**キャッシュライン分割はロードのペナルティーとなります
- **どのように検出するか：**メモリー依存のサブカテゴリー、測定基準：*"Split Loads"*、*"Split Stores"* を調査します
- **何を行うか：**
  - ホットスポットでこの測定基準がハイライトされている場合、ソースコード・レベルで測定基準を識別して調査します。分割ロードイベント、"MEM_UOP_RETIRED.SPLIT_LOADS_PS" は、分割ロードの発生後に実行された次の命令にタグ付けされています。分割ストアのレシオが .01 より大きい場合、調査する必要があります
  - この問題を解決するには、データのアライメントを確実にします。特に、256 ビットの AVX ストア操作のミスアラインに注目します

その他のデータアクセスの問題：4K エイリアス

- **なぜ重要なのか：**エイリアスの競合により、ロードを再発行しなければいけなくなります。
- **どのように検出するか：**メモリー依存のサブカテゴリー、測定基準：*“4K Aliasing”* を調査します
- **何を行うか：**
  - ホットスポットでこの測定基準がハイライトされている場合、ソースコード・レベルで測定基準を識別して調査します
  - ロードのアライメントを変更することで、この問題を解決します。入力と出力バッファー間のオフセットを変更することで（可能であれば）、データを 32 バイトにアライメントするか、32 バイトにアライメントされていないメモリーへのアクセスに 16 バイトのメモリーアクセスを使用します

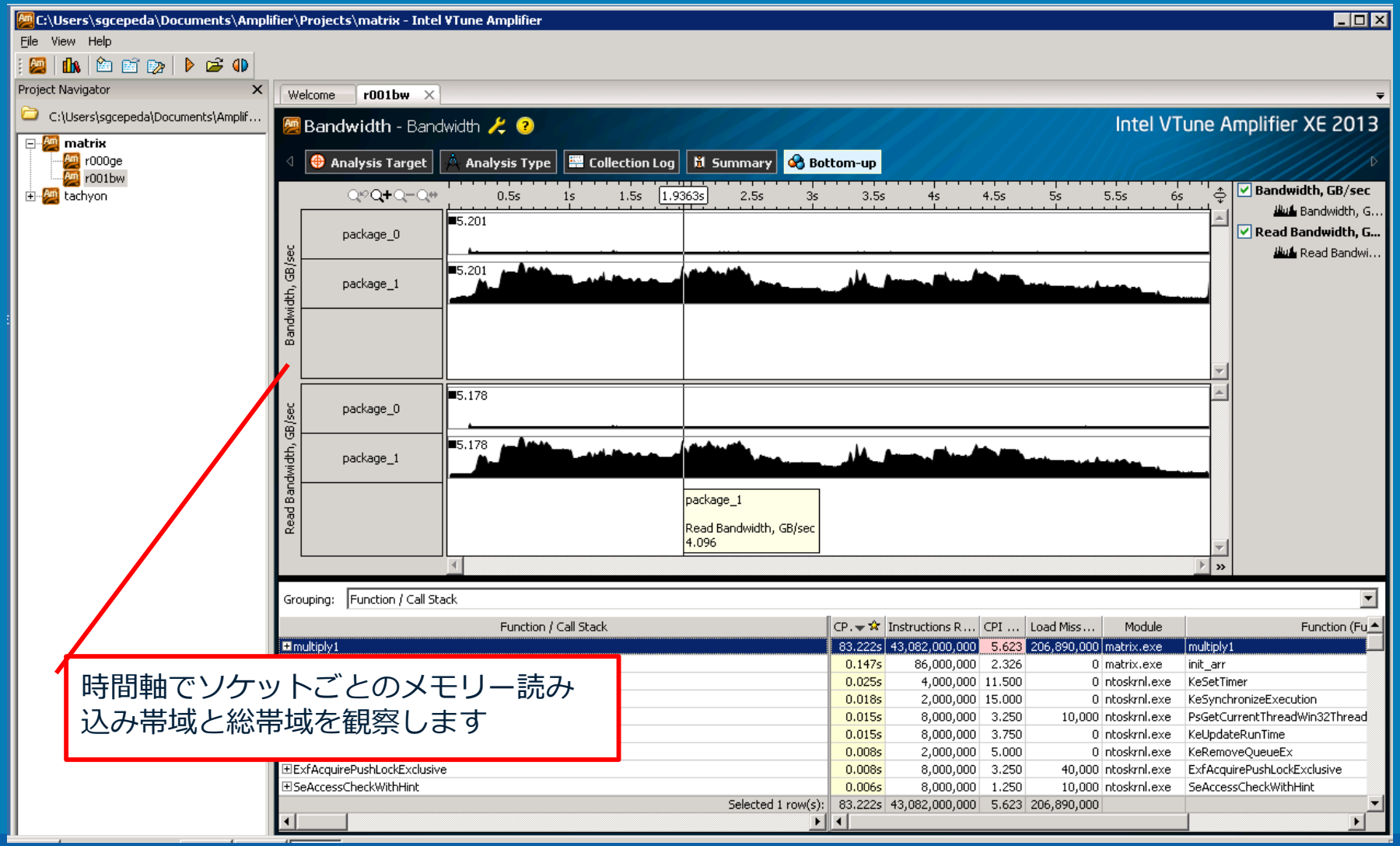
その他のデータアクセスの問題：DTLB ミス

- **なぜ重要なのか：**最初のレベルの DTLB ロードミス（STLB にヒット）は、レイテンシーのペナルティーがかかります。第 2 レベルのミスではページウォークが必要となり、アプリケーションのパフォーマンスに影響します
- **どのように検出するか：**メモリー依存のサブカテゴリー、測定基準：*"DTLB Overhead"*、*"DTLB Store Overhead"* を調査します
- **何を行うか：**
  - ホットスポットでこの測定基準がハイライトされている場合、ソースコード・レベルで測定基準を識別して調査します
  - この問題を解決するには、より優れたメモリー割り当て機能を利用したり、プロファイル・ガイド最適化により、ターゲットデータの局所性を TLB のサイズに合わせるか、仮想化システムでは拡張ページテーブル（EPT)を使用するか、ラージページを使用するか（データベースやサーバー・アプリケーションのみで）、データの局所性を高めます

ボーナス問題：メモリー帯域幅の制限

- **なぜ重要なのか：**バンド幅のボトルネックは、キャッシュミスが発生した時にレイテンシーを増加させます
- **どのように検出するか：**"Bandwidth" のプロファイルを調査します
- **何を行うか：**
  - プラットフォームのソケットごとにメモリーの理論的な最大帯域幅を GB/秒で計算します:
    (<*MT/秒*> * 8 バイト/クロック * <チャンネル数>) / 1000
  - アプリケーションを実行してバンド幅を計測します。ソケットごとの総帯域幅が 75% 以上である場合、アプリケーションには（高い）ロード・レイテンシーがある可能性があります
  - 適切であれば、システムチューニングや調整を行います（メモリー DIMM のアップグレードやバランス調整、ハードウェア・プリフェッチャーの無効化）
  - 可能であれば帯域幅の利用を軽減します：非効率な SW プリフェッチの削除、データ領域や共有を減らすアルゴリズムの変更、データ更新の軽減、ソケットに分散するメモリーアクセス・バランスの調整などが考えられます

帯域幅解析



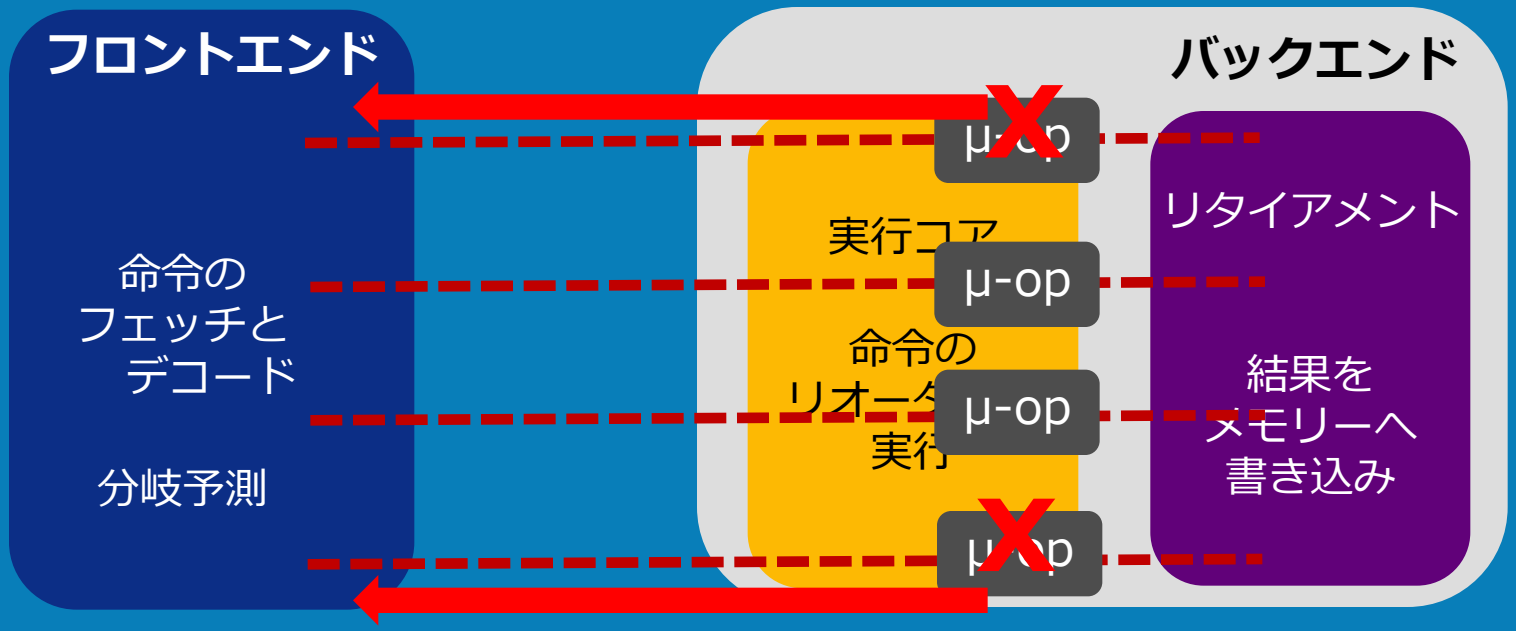
時間軸でソケットごとのメモリー読み込み帯域と総帯域を観察します

投機の問題のカテゴリーをチューニング

投機が行われる場合：

- マイクロ・オペレーションは、それがリタイアするかどうかにかかわらず実行が許可されます
- これは、アウト・オブ・オーダー・パイプラインで高い命令レベルの並列性をもたらします

投機の問題の概要



マイクロ・オペレーション(μ-op) が
バックエンドから削除され、リタイアしない場合に発生

分岐予測ミス

- **なぜ重要なのか：**誤って予測された分岐は、無駄な処理や命令スタベーション（新しい命令がフェッチされるまで待機するため）により、パイプラインの効率を低下させます
- **どのように検出するか：**General Exploration プロファイル、測定基準：*"Branch Mispredict"* を調査します
- **何を行うか：**
  - ホットスポットでこの測定基準がハイライトされている場合、ソースコード・レベルで測定基準を識別して調査します
  - コンパイラー・オプションやプロファイル・ガイド最適化によってコード生成を改善します
  - 分岐文において、もっとも分岐するターゲットが主な命令フローとなるように手動で調整します

マシンクリア

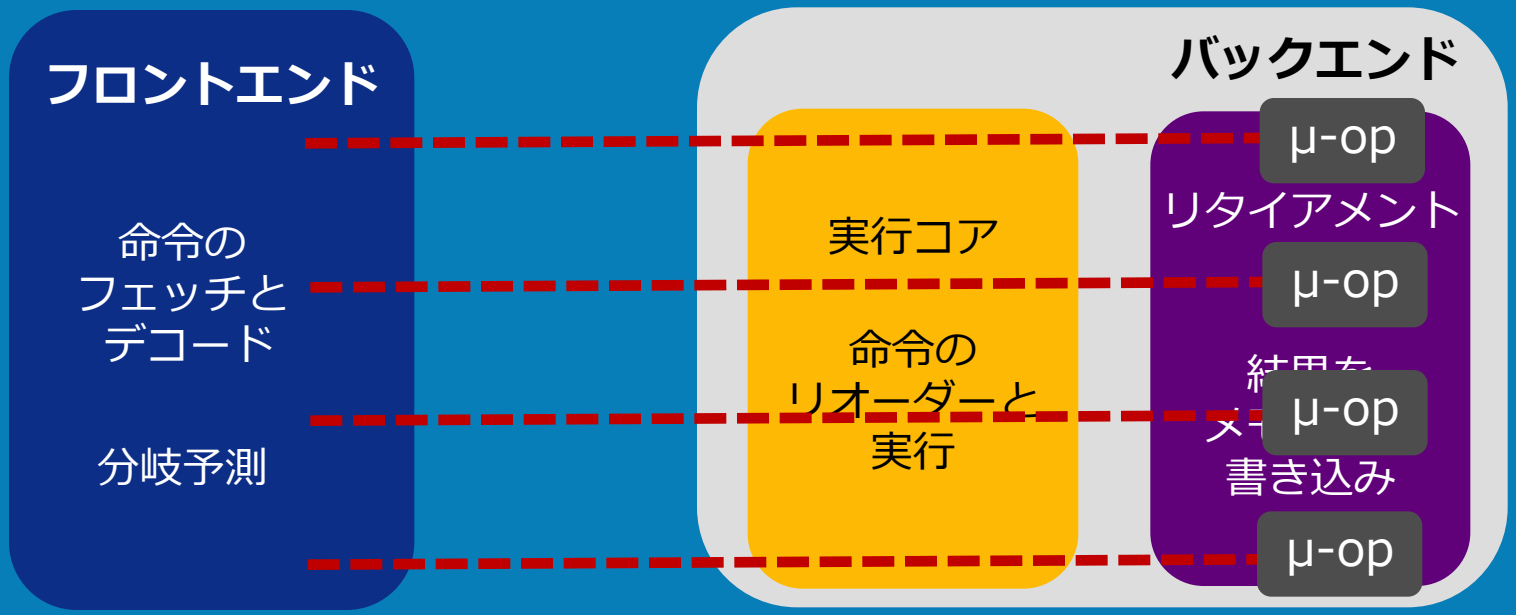
- **なぜ重要なのか：**マシンクリアは、パイプラインをフラッシュし、ストアバッファーを空にするため、大幅なレイテンシーのペナルティーとなります
- **どのように検出するか：**General Exploration プロファイル、測定基準：*"Machine Clears"* を調査します
- **何を行うか：**
  - ホットスポットでこの測定基準がハイライトされている場合、幾つかのイベントを使用して原因を特定します：
    - "MACHINE_CLEARS.MEMORY_ORDERING" イベントが多い場合、ソースコード・レベルで調査します。これは、4K エリアス競合やロック競合によって引き起こされます
    - "MACHINE_CLEARS.SMC" イベントが多い場合、自己修正コードによる原因です。これは避けるべきです

リタイアメント・カテゴリーのチューニング

リタイアメントは：

- マイクロ・オペレーション実行の完了を示します
- マイクロ・オペレーションが命令を構成する最後のオペレーションであれば、命令実行の完了を意味します
- 命令の実行結果は、アーキテクチャー状態（キャッシュ、メモリーなど）にコミットされます

リタイアの概要



パイプライン・スロットがマイクロ・オペレーション(μ-op)で満たされた時に発生します。これは望ましいカテゴリーです。しかし、いくつかの問題もあります

インテル® VTune™ Amplifier XE における リタイア階層のブレークダウン

Filled Pipeline Slots		Unfilled Pipeline Slots (Stalls)	
Retiring	Bad Speculation	Back-end Bound	Front-end Bound
0.803	0.006	0.140	0.051
0.208	0.010	0.730	0.053
0.105	0.072	0.563	0.260
0.437	0.000	0.540	0.224
0.052	0.040	0.706	0.202
0.132	0.000	0.550	0.318

Retiring	
General Retirement	Microcode Sequencer
0.808	0.001
0.175	0.043
0.107	0.070
0.075	0.161
0.000	0.167
0.000	0.180

最良のケースである “General Retirement” とマイクロコード・シーケンサーから生成されたマイクロ・オペレーションのリタイアである “Microcode Sequencer” で区分されるリタイアスロットの比率を見るには、“Retiring” を展開表示します

マイクロコード・アシスト

- **なぜ重要なのか：**マイクロコード・シーケンサーからのアシストには、長いレイテンシーのペナルティーが伴います
- **どのように検出するか：**General Exploration プロファイル、測定基準 : "*Microcode Sequencer*" を調査します
- **何を行うか：**
  - ホットスポットでこの測定基準がハイライトされた場合、他のアシストイベントを使用して原因を特定するため再度サンプリングを行います
  - もし、"FP_ASSISTS.ANY / INST_RETIRED.ANY" がかなり多いようであれば、浮動小数点演算のデノーマルを調べてください。SSE/AVX 命令を使用している場合、FTZ と（もしくは） DAZ を有効にするか、演算をスケールアップもしくはダウンすることで解決できます
  - もし、 "((OTHER_ASSISTS.AVX_TO_SSE_NP*75) / CPU_CLK_UNHALTED.THREAD)" もしくは、"((OTHER_ASSISTS.SSE_TO_AVX_NP*75) / CPU_CLK_UNHALTED.THREAD) " が、.1 より大きければ SSE と AVX コード間の移行を減らします。詳細は http://software.intel.com/en-us/articles/avoiding-avx-sse-transition-penalties を参照してください

"ハードウェア上のソフトウェア" のチューニング手順

各ホットスポットで

- 効率を判断
  - > もし十分でなければ：
    - 主なボトルネックを特定
    - 非効率であるアーキテクチャー上の理由を見つけます
    - 問題を最適化します

繰り返します!

目立ったホットスポットが無くなるまで繰り返すと、
リタイアのパイプライン・スロットは期待どうりとなるでしょう

ありがとうございます！
より詳しい情報について：

インテル® VTune™ Amplifier XE 向けビデオ、フォーラムおよびリソース：
http://software.intel.com/en-us/intel-vtune-amplifier-xe/#pid-3659-760/

インテル® 64 アーキテクチャーおよび IA-32 アーキテクチャー・ソフトウェア開発マニュアル：
http://www.intel.com/products/processor/manuals/index.htm

他のマイクロ・アーキテクチャー向けのインテル® VTune™ Amplifier XE のチューニング・ガイド：
http://software.intel.com/en-us/articles/processor-specific-performance-analysis-papers

統合グラフィック・コントローラーの最適化：
www.intel.co.jp/jp/software/products

法律的な免責条項

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスも許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品適格性、あらゆる特許権、著作権、その他知的財産権の非侵害性への保証を含む) に関してもいかなる責任も負いません。

「ミッション・クリティカルなアプリケーション」とは、インテル製品の故障の結果として、直接的または間接的に、人体への危害または死亡事故が発生するおそれがあるアプリケーションです。かかるミッション・クリティカルなアプリケーションの用途でインテル製品を購入または使用した場合、かかるミッション・クリティカルなアプリケーションに関連する製造物責任、人体への危害、または死亡事故の申立から直接的または間接的に生じる、すべての申立、費用、損害、支出、妥当額の弁護士費用に対して、かかる申立がインテル製品またはその部品の設計、製造、または警告に関してインテルまたはインテルの外部委託業者の過失があったと主張する場合であっても、インテルおよびその子会社、外部委託業者および関連会社、取締役、役員、各会社の従業員を免責、保護するものとします。

インテル製品は、予告なく仕様や説明が変更されることがあります。機能または命令の一覧で「留保」または「未定義」と記されているものがありますが、その「機能が存在しない」あるいは「性質が留保付である」という状態を設計の前提にしないでください。これらの項目は、インテルが将来のために留保しているものです。インテルが将来これらの項目を定義したことにより、衝突が生じたり互換性が失われたりしても、インテルは一切責任を負いません。この情報は予告なく変更されることがあります。この情報だけに基づいて設計を最終的なものとしないでください。

本資料で説明されている製品には、エラッタと呼ばれる設計上の不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。現在確認済みのエラッタについては、インテルまでお問い合わせください。

本プレゼンテーションに記載されたインテルの製品計画は、インテルの POR 製品ロードマップではありません。インテルの現在の POR 製品ロードマップをご希望の方は、インテルの担当者までお問い合わせください。

インテル・プロセッサー・ナンバーはパフォーマンスの指標ではありません。プロセッサー・ナンバーは同一プロセッサー・ファミリー内の製品の機能を区別します。異なるプロセッサー・ファミリー間の機能の区別には用いません。詳細については、
http://www.intel.co.jp/jp/products/processor_number / を参照してください。

最新の仕様をご希望の場合や製品をご注文の場合は、お近くのインテルの営業所または販売代理店にお問い合わせください。

本書で紹介されている注文番号付きのドキュメントや、インテルのその他の資料を入手するには、1-800-548-4725 (アメリカ合衆国) までご連絡いただくか、http://www.intel.com/design/literature.htm (英語) を参照してください。

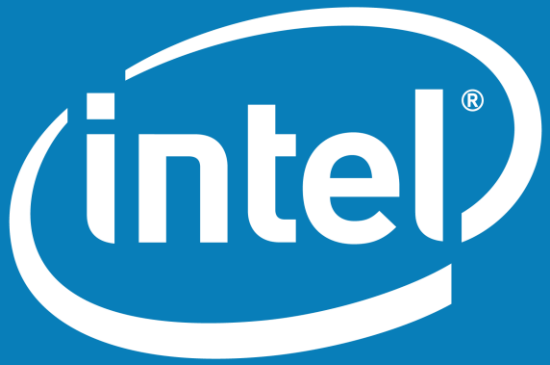
Haswell、Ivy Bridge、Sandy Bridge、Westmere、Nehalem、Merom、Yonah、Banias および記載されているその他の開発コード名は、まだ発表されていない開発中の製品を識別するためにインテル社内で使用されているものです。顧客、ライセンス契約者、またはその他の第三者が任意の製品またはサービスの広告、販売促進、またはマーケティングにコード名を使用することは、インテルに認定されていません。このような目的でインテルの社内コード名を使用する場合は、利用者の責任において行ってください。

Intel、インテル、Intel ロゴ、Core、VTune、Ultrabook、Sponsors of Tomorrow は、アメリカ合衆国および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、一般に各社の表示、商標または登録商標です。

最適化に関する免責条項

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


intel