インテル® プロセッサーのサイクル・アカウンティング

ソフトウェア＆ソリューションズ統括部

ソフトウェア製品部

Rev Aug 15, 2008

目的

インテル® Core™ マイクロアーキテクチャー・プロセッサー搭載のシステムで実行するソフトウェアのマイクロアーキテクチャー上のボトルネックを識別する

コースの内容

パフォーマンス解析とイベントの基本

インテル® Core™ アーキテクチャー・プロセッサーのボトルネックを識別するイベント

まとめ

方法論の概要

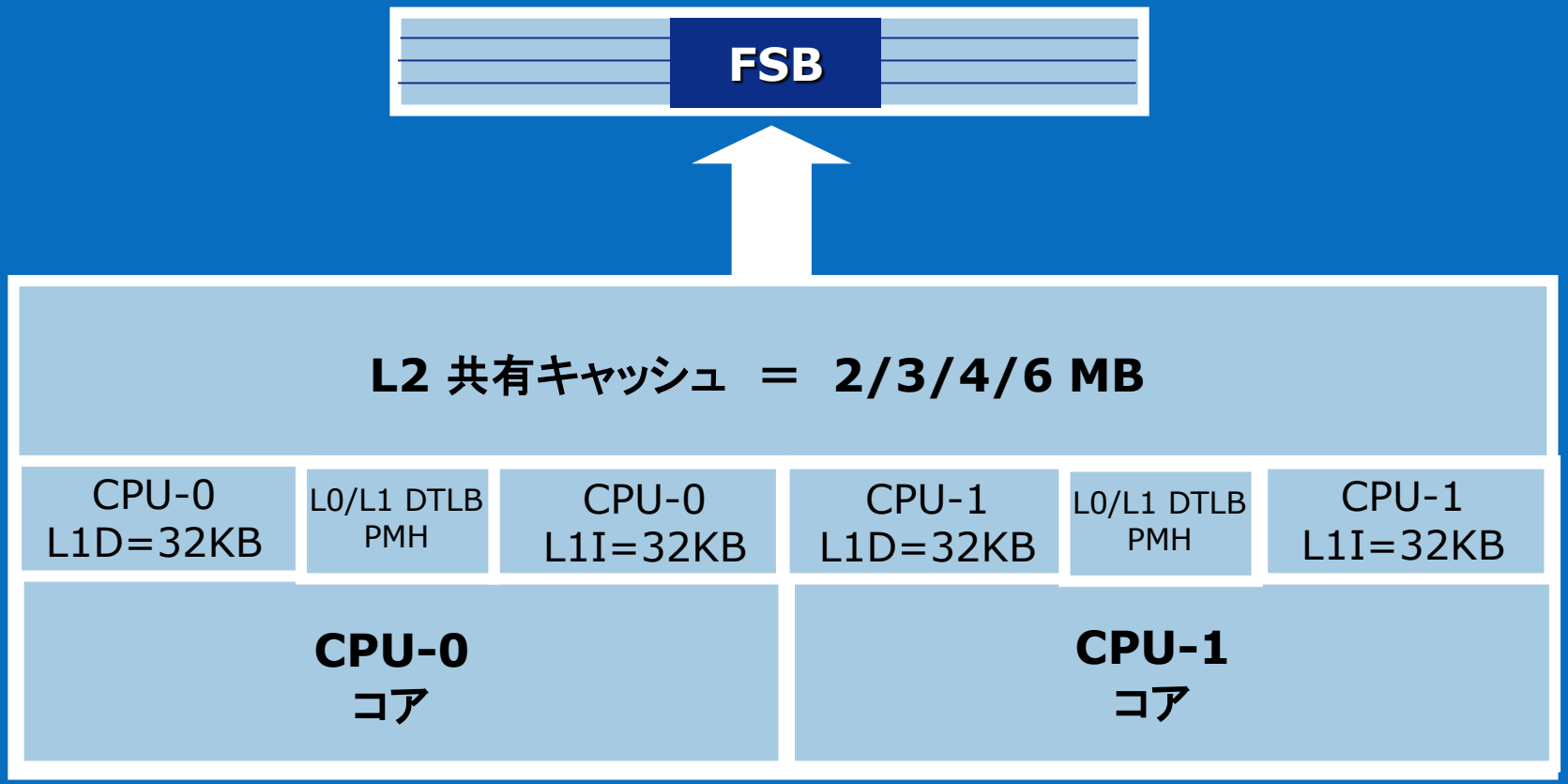
- x86 プロセッサーにおける伝統的なパフォーマンス・チューニングは、命令のリタイアに注目する
- OOO エンジンの実行動作は不透過で、予測することが困難である

そのため命令のリタイアに注目することは最善の方法ではない

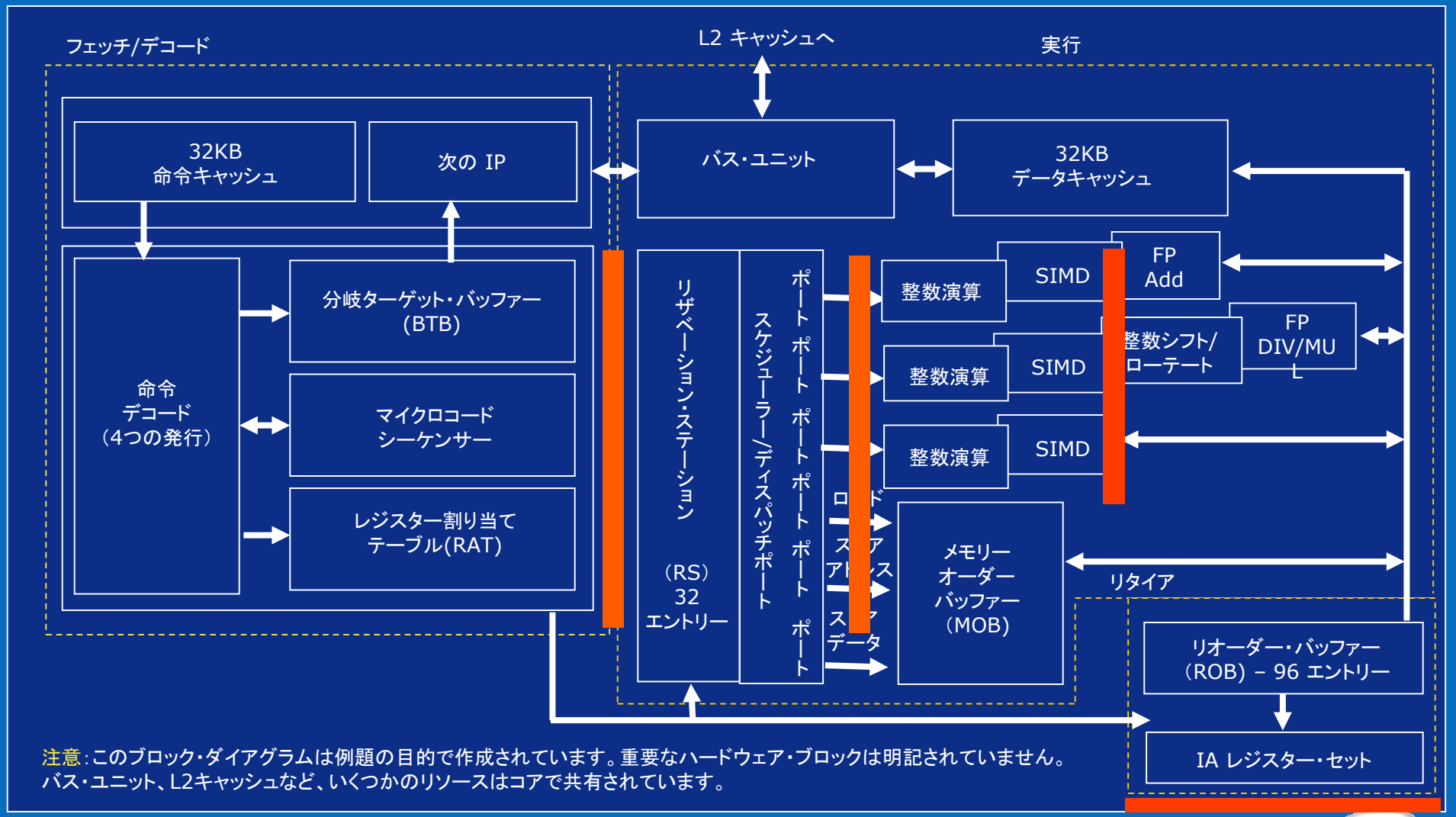
サイクル分析による方法論

- この手法では、4 フェーズのパフォーマンス・イベント測定によって、主要なパイプステージまたはマイクロアーキテクチャー・サブシステムとコードの相互作用を特性評価する
- この方法による最適化は 2 つの要素からなる
  - 命令カウントを最小限にする
    - “木構造”の最小化
  - 理想的な実行ルートからの逸脱を最小限にする
    - 一般的に“ストール・サイクル”として考えられる
  - 両者を同一に扱うことは危険である

新世代マイクロアーキテクチャー
インテル® Core™ マイクロアーキテクチャー・プロセッサー

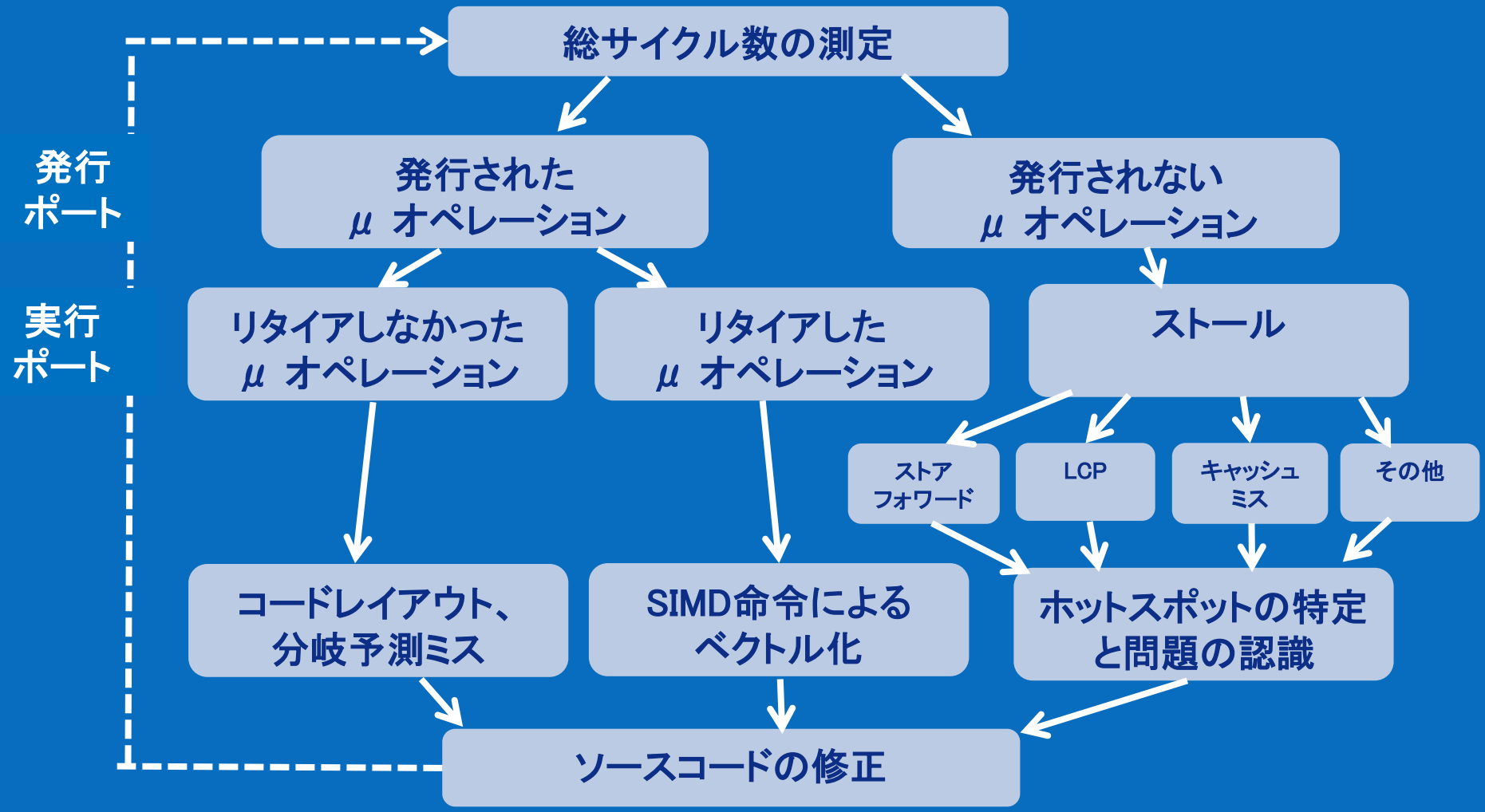


インテル® Core™ マイクロアーキテクチャー ブロック・ダイアグラム



注意：このブロック・ダイアグラムは例題の目的で作成されています。重要なハードウェア・ブロックは明記されていません。
バス・ユニット、L2キャッシュなど、いくつかのリソースはコアで共有されています。

パフォーマンス・イベントのドリルダウン



総サイクル数の測定

一般的なパフォーマンス・チューニングの状況では、総サイクル数は CPU_CLK_UNHALTED.CORE イベントによって測定できる

CPU_CLK_UNHALTED.NO_OTHER（コアがアクティブでもう一方のコアが HALT 中のバスサイクル）イベントをバスの周波数で割ることで、1つのコアがどれくらいバスを占有しているか解る

CPU_CLK_UNHALTED.CORE イベントは、プロセッサーが HALT（停止）状態ではないサイクルを測定する

CPU_CLK_UNHALTED.NO_OTHER を監視することで、アプリケーションやシステムが利用するバスの並列利用度が解る

発行ポートでのサイクル構成

リザベーション・ステーション（RS）は、プログラムが処理を進行するためマイクロオペレーション（μOP）をディスパッチする

総サイクルは、2 つの排他的なコンポーネントで構成される：

総サイクル = Cycles_not_issuing_uops + Cycles_issuing_uops

Cycles_not_issuing_uops は、RS が実行されるマイクロオペレーションを発行していないサイクル数を示す。アウトオブオーダー・エンジンがストールしたサイクル数（Cycles_stalled）を表す

Cycles_issuing_uops は、RS が実行されるマイクロオペレーションを発行しているサイクル数を示す。正しく実行されたコードパスまたはスペキュレーティブなコードパスのマイクロオペレーションが含まれる

実行ポートでのサイクル構成

アウトオブオーダー・エンジンは、複数のマイクロオペレーション（μOP）を並行して実行できる複数の実行ユニットを備えている

1 つの実行ユニットがストールしても、プログラムの実行がストールするとは限らない。このドリルダウン手法では、プログラムの実行状況を近似的に示すサイクルを参照する

関連する指標は、

Cycles_stalled （ストールしたサイクル）
Cycles_not_retiring_uops （ストールしたサイクル）
Cycles_retiring_uops（リタイアした命令のサイクル）

リタイアイベントと非リタイアイベント

リタイアしたイベントとは、マシンステートを確定した命令によるイベントのみを含む

- INST_RETIRED.ANY（リタイアした命令の数）イベントを測定している場合、予測をミスした実行パスで発生したロードはカウントされない

非リタイアイベントは、インテル® Core™ マイクロアーキテクチャーにおけるアウトオブオーダーの推測による流れで発生する

チューニングの指標

総サイクルを基準として、Cycles_non_retiring_uops、Cycles_stalled、Cycles_retiring_uops の3 つの指標を評価すれば、チューニング作業に役立つ以下の指針が得られる

- Cycles_non_retiring_uops が大きい場合は、コードのレイアウトに焦点を当て、分岐の予測ミスを減らすことが重要である。

- Cycles_non_retiring_uops と Cycles_stalled がいずれも小さい場合は、パフォーマンス・チューニングでベクトル化などの手法を重視し、Hotspot 関数のリタイアメント・スループットを向上させるべきである

- Cycles_stalled が大きい場合は、マイクロアーキテクチャー・パイプラインのさらに深い個所に潜むボトルネックを発見するために、さらなるドリルダウンが必要である

パフォーマンス・ストールのドリルダウン

個々のストレスポイント（ストール）が原因で失われたサイクル数を合計する手法は、コードをチューニングして各ストレスポイントのパフォーマンスへの影響を解決する際に有益である

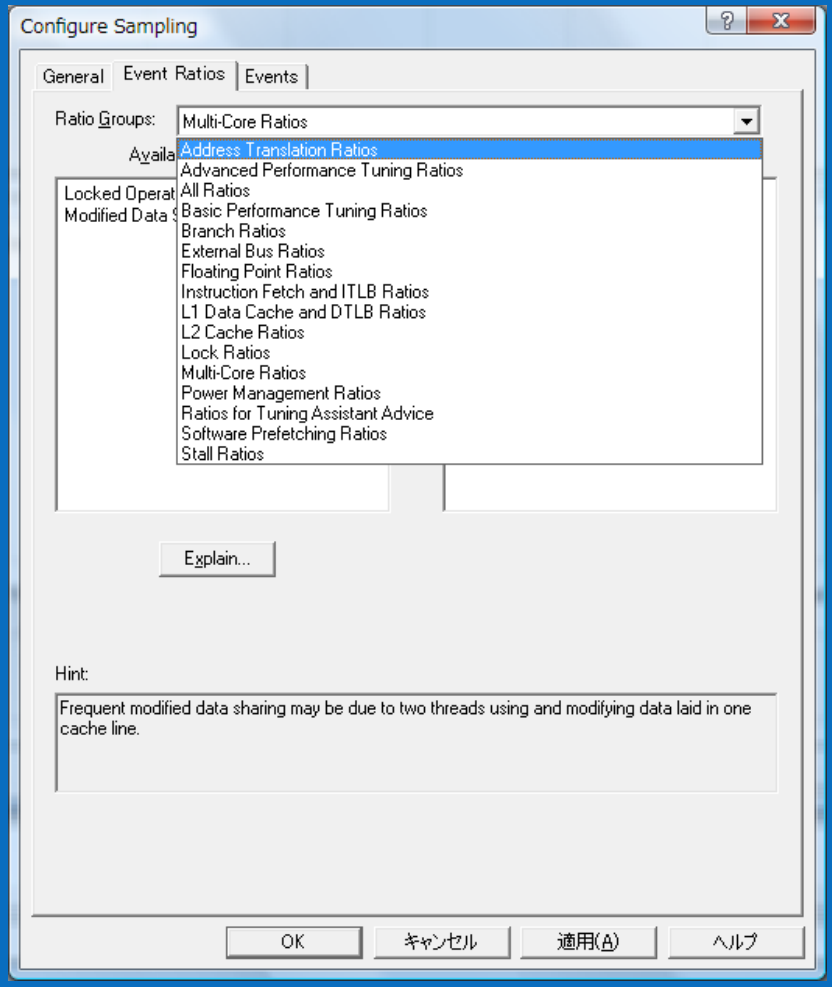
マイクロアーキテクチャーにおける一般的なストレスポイント：

L2 ミスの影響 （数百サイクル）
L2 ヒットの影響 （14 もしくは 15 サイクルのアクセス・レイテンシー）
L1 DTLB ミスの影響 （ルックアップ・ミスは約 10 サイクル）
LCP の影響 （6 サイクルのコスト）
ストア・フォワーディングによるストールの影響（5 から 20 サイクルのコスト）

イベントとイベント比率

一般的なプロセッサーのパフォーマンス要因を計算

インテル® VTune™
パフォーマンス・アナライザーでは、各プロセッサーのアーキテクチャーごとのイベント比率が提供される



基本イベント

イベント	P	説明	イベント	P	説明
CPU_CLK_UNHALTED			BUS_DRDY_CLOCKS.ALL_AGENTS		全てのBUSYバス・サイクル
INST_RETIRED_ANY_P	P		BUS_DRDY_CLOCKS.THIS_AGENT		書き込みによる全てのBUSYバス・サイクル
INST_RETIRED_LOADS			MEM_LOAD_RETIRED.L2_LINE_MISS	P	L2の要求ミス
INST_RETIRED_STORES			MMX2_PRE_MISS.T1		L1キャッシュへのSWプリフェッチ
BUS_TRANS_ANY		全てのバス・トランザクション	MMX2_PRE_MISS.T2		L2キャッシュへのSWプリフェッチ
BUS_TRANS_MEM		メモリーへのバス・トランザクション	MMX2_PRE_MISS.STORES		実行された非テンポラルストア
BUS_TRANS_BURST		ラインのメモリー書き込み	L2_LINES_IN.SELF.DEMAND		SWプリフェッチによるL2へのライン転送
BUS_TRANS_BRD		ラインのメモリーからの読み込み	L2_LINES_IN.SELF.PREFETCH		HWプリフェッチによるL2へのライン転送
BUS_TRANS_WB		ライトバック（NTライトはなし）	L2_LINES_OUT.SELF.DEMAND		要求されたL2の追い出し
BUS_TRANS_RFO		RFOへのライン転送（HWプリフェッチではない）	L2_LINES_OUT.SELF.PREFETCH		HWプリフェッチによるL2の追い出し

特定のイベント比率は幾つかのイベントを組み合わせた計算式で求められる：

メモリーバンド幅 = 64 * Bus_Trans_Mem * 周波数 / Cpu_Clk_Unhalted

CPI 値に注目することから始める

CPU_CLK_UNHALTED.CORE（CPU が HALT 中では無いコアサイクル）イベントで CPU サイクルを測定

CPU_CLK_UNHALTED.CORE /プロセッサーの周波数 ＝ 1秒あたりの回数

INST_RETIRED.ANY（リタイアした命令の数）＝プロセッサー・ステートが確定する（完全に実行された）命令の数

CPI（命令あたりのリタイアサイクル数）＝
CPU_CLK_UNHALTED.CORE / INST_RETIRED.ANY

CPI の値が高い場合、最適化の余地がある

**インテル® Core™ アーキテクチャーの
プロセッサーの最良値は 0.25**

演習 1 :

マルチコア・システムでの CPI の計算

インテル® VTune™ パフォーマンス・アナライザーを使用して、
マルチスレッド・アプリケーションの CPI を計算する

演習テキスト:インテル® Core™ マイクロアーキテクチャーのパフォーマンス・カウンター
演習 1: CPI の計算を参照

コースの内容

パフォーマンス解析とイベントの基本

インテル® Core™ アーキテクチャー・プロセッサーのボトルネックを識別するイベント

まとめ

ストールと不完全な実行

- ストール・サイクルは実行の不完全さを表す
  - プロセッサーのアーキテクチャーに依存するストールは、アーキテクチャー・イベントを監視することで識別できる
  - ストールと IP の関連およびアーキテクチャー・イベントは最適化の指標となる
- OOO エンジンには 4 つの代表的なストールがある
  - 実行ストール
    - 入力やスコアボード待ち、L2 ミス、BW、DTLB、その他
  - 分岐予測ミスによるパイプラインのフラッシュ
  - FE (フロント・エンド) ストール
    - 実行ステージで命令が足りなくなる
  - リタイアしない命令によって CPU サイクルが浪費される

x86 サイクル計算とソフトウェア最適化

- CPU_CLK_UNHALTED = “ストール” + NON_RET_DISPATCH + RET_DISPATCH

代表的なストールの排除

分岐予測ミスを減らすPGO

命令カウントを減らすため最適化を向上させ、命令レベルの並列性（ILP）を増すためループを分割する

RESOURCE_STALLS.BR_MISS_CLEAR
イベントでパイプライン・フラッシュによるストールを算出

実行ステージでの有効性を計測

OOO エンジンはリザベーション・ステーション（RS）から実行ユニットへの命令ディスパッチを行う

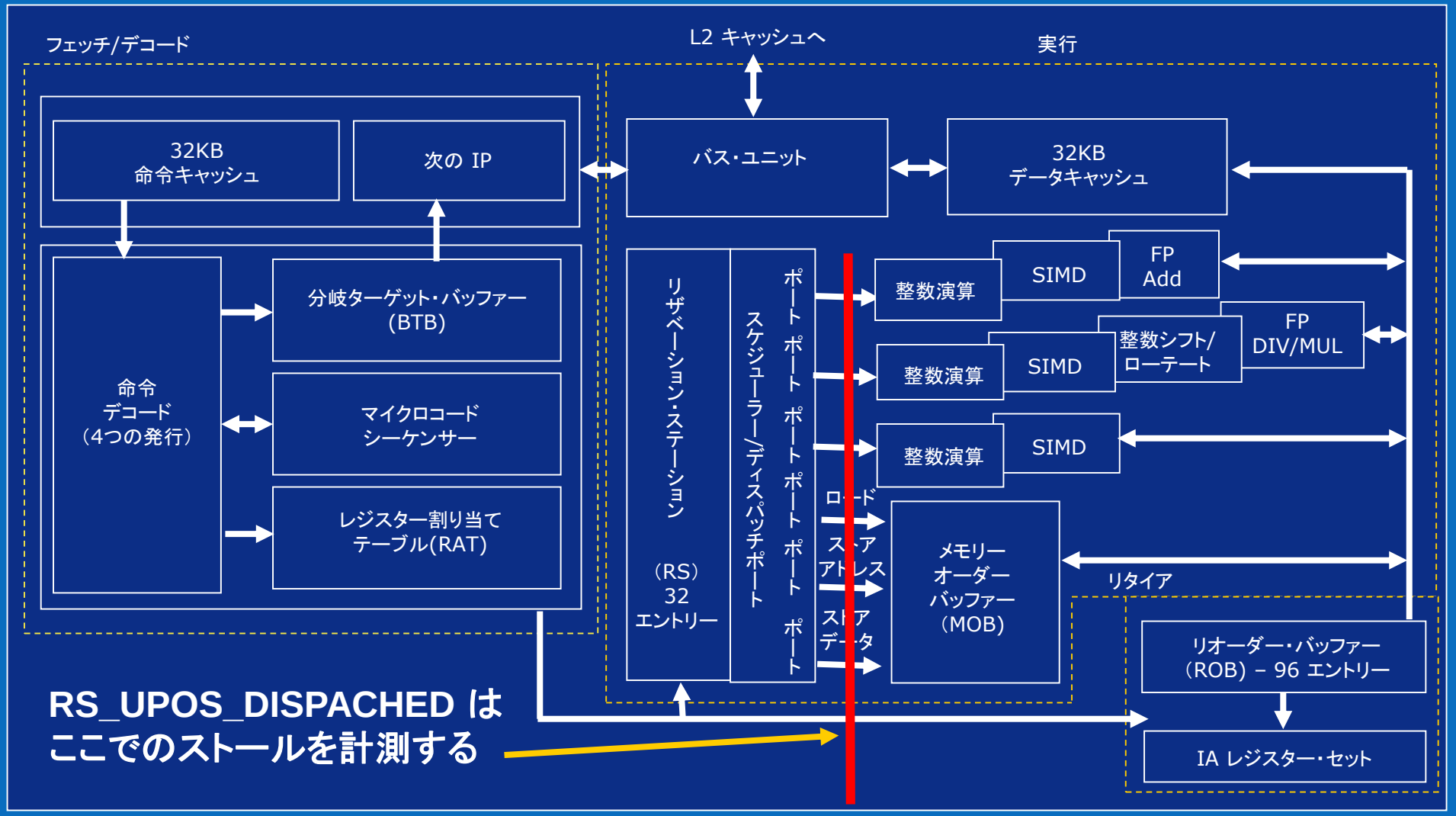
- ディスパッチされた命令は入力ソースが利用可能になるまで待つ

- RS_UOPS_DISPATCHED は、各 CPU サイクルで RS からディスパッチされたマイクロ・オペレーションの数をカウントする

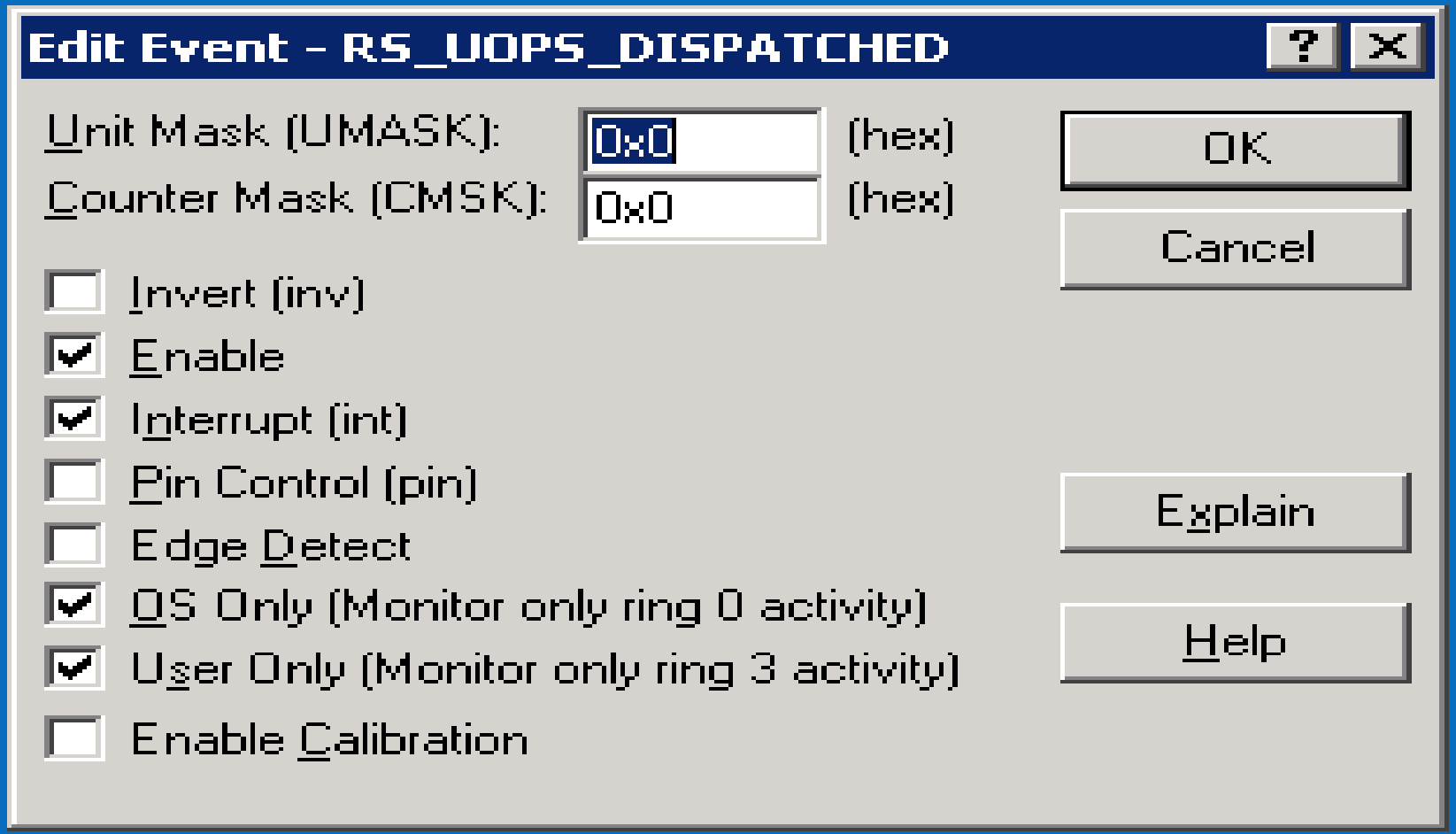
最も重要な PMU* イベント

*** Performance Monitoring Unit (PMU)**

実行時に何が起こっているか

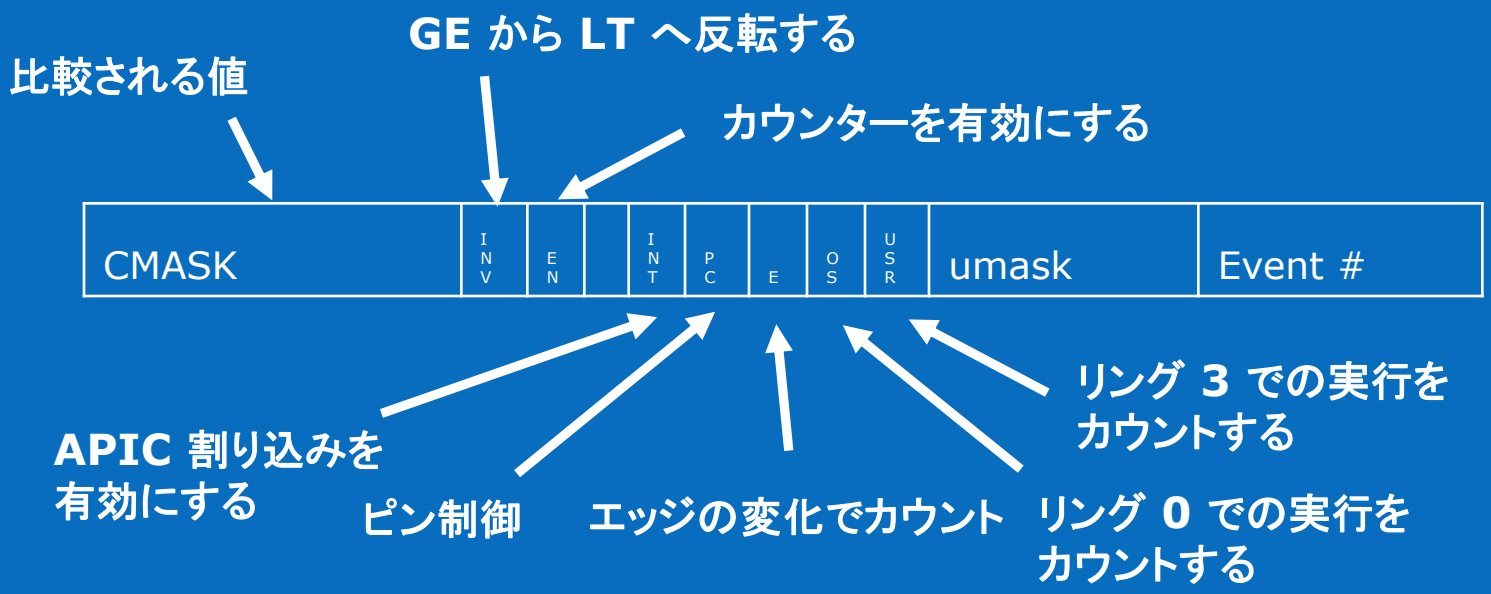


VTune™ アナライザーのイベント編集



インテル® ソフトウェア開発製品

PMU のいくつかの機能



RS_UOPS_DISPATCHED で、ストールの発生をカウントするため、CMASK = 1 、INV = 1 に設定

VTune™ アナライザーは上記の定義済イベントを持つ RS_UOPS_DISPATCHED.CYCLES_NONE

総クロックサイクルの分類

RS_UOPS_DISPATCHED.CYCLES_ANY （定義済イベント）

RS_UOPS_DISPATCH: cmask = 1　　ディスパッチされたサイクル数

RS_UOPS_DISPATCH: cmask = 1 : inv = 1　　ディスパッチされなかった（ストールした）サイクル数

RS_UOPS_DISPATCHED.CYCLES_NONE （定義済みイベント）

CPU_CLK_UNHALTED は OOO エンジンでのストールと実行に区分できる

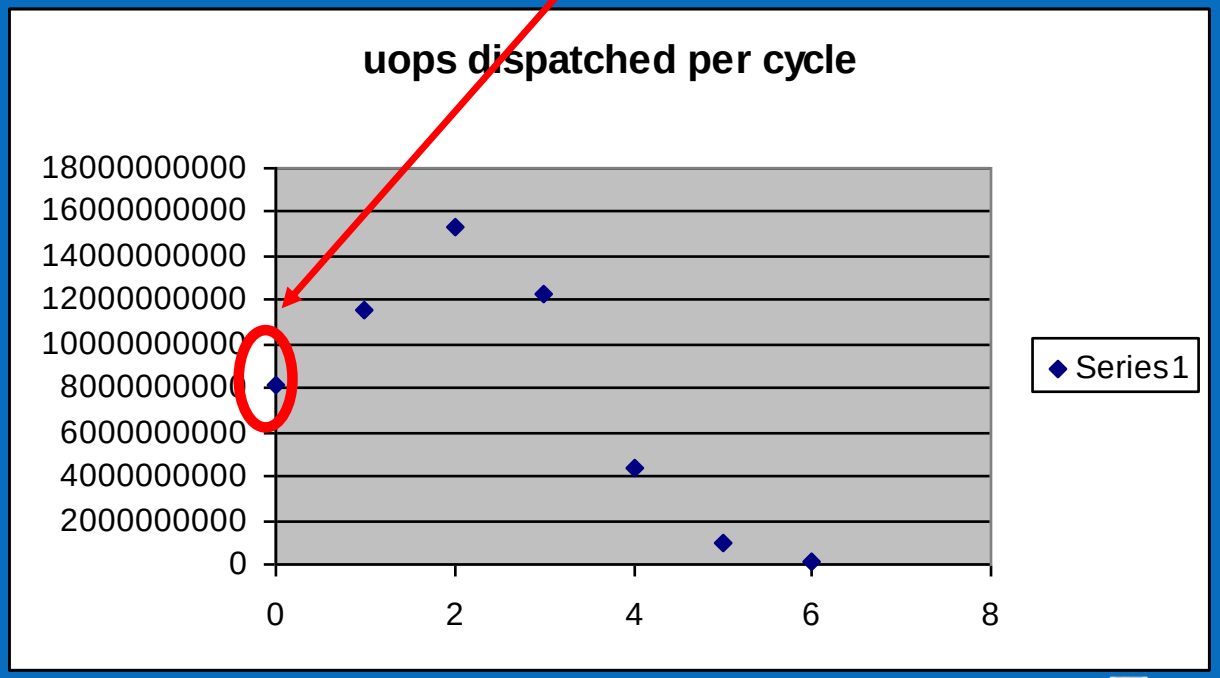
RS_UOPS_DISPATCHED.CYCLES_NONE が検知されたらその要因を分類する

UOP / サイクルの分布

RS_UOPS_DISPATCHED:cmask=1:inv=1
RS_UOPS_DISPATCHED:cmask=2:inv=1
RS_UOPS_DISPATCHED:cmask=3:inv=1
RS_UOPS_DISPATCHED:cmask=4:inv=1
RS_UOPS_DISPATCHED:cmask=5:inv=1
RS_UOPS_DISPATCHED:cmask=6:inv=1
RS_UOPS_DISPATCHED:cmask=7:inv=1

Subtract the N-1 value

ストール
サイクル



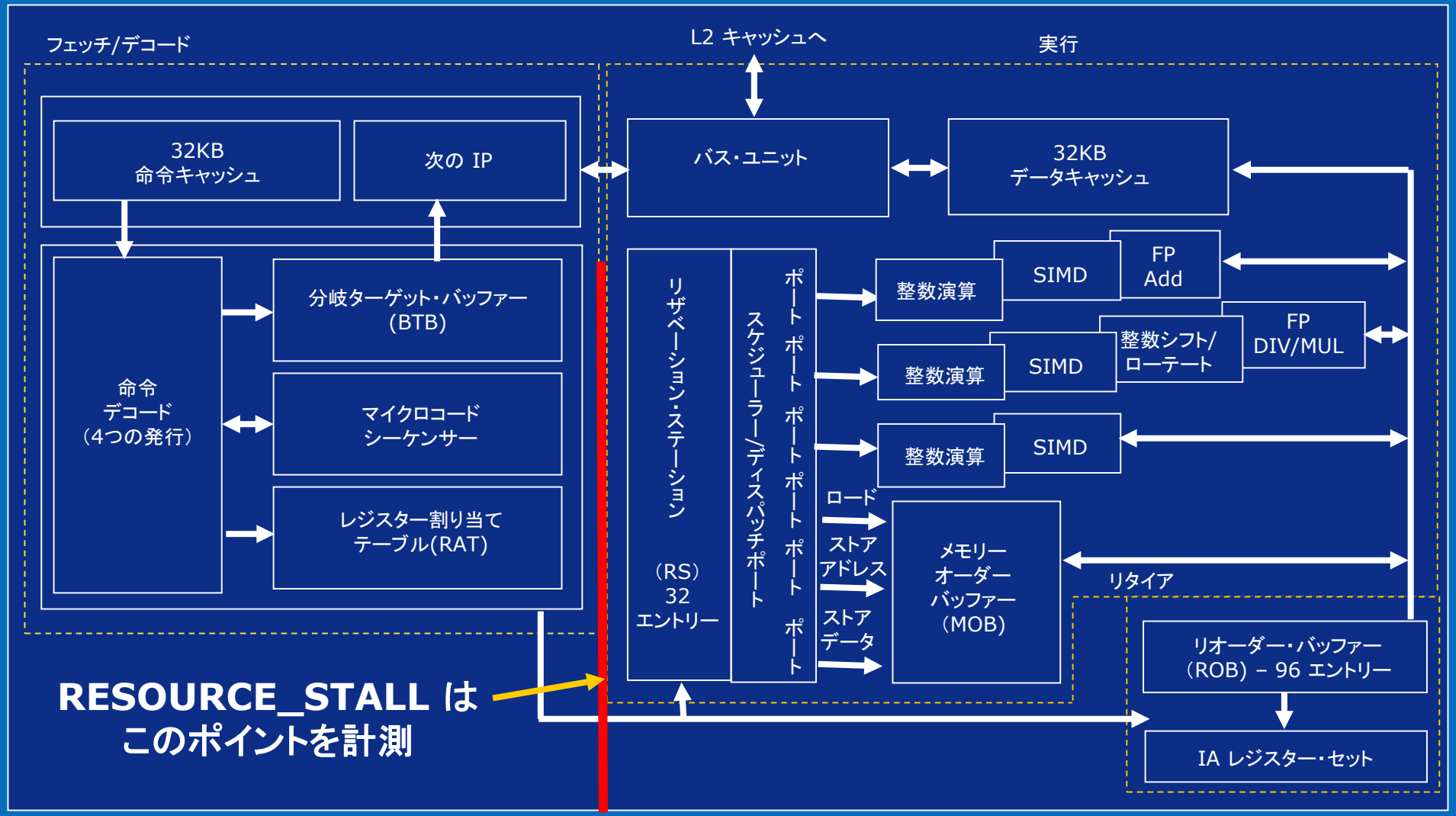
命令レベルの
並列性の分布

例:

```
for(i=0;i<len;i++)
 a[i] = exp(b[i]);
```


インテル® ソフトウェア開発製品

アーキテクチャー・ブロック・ダイアグラム

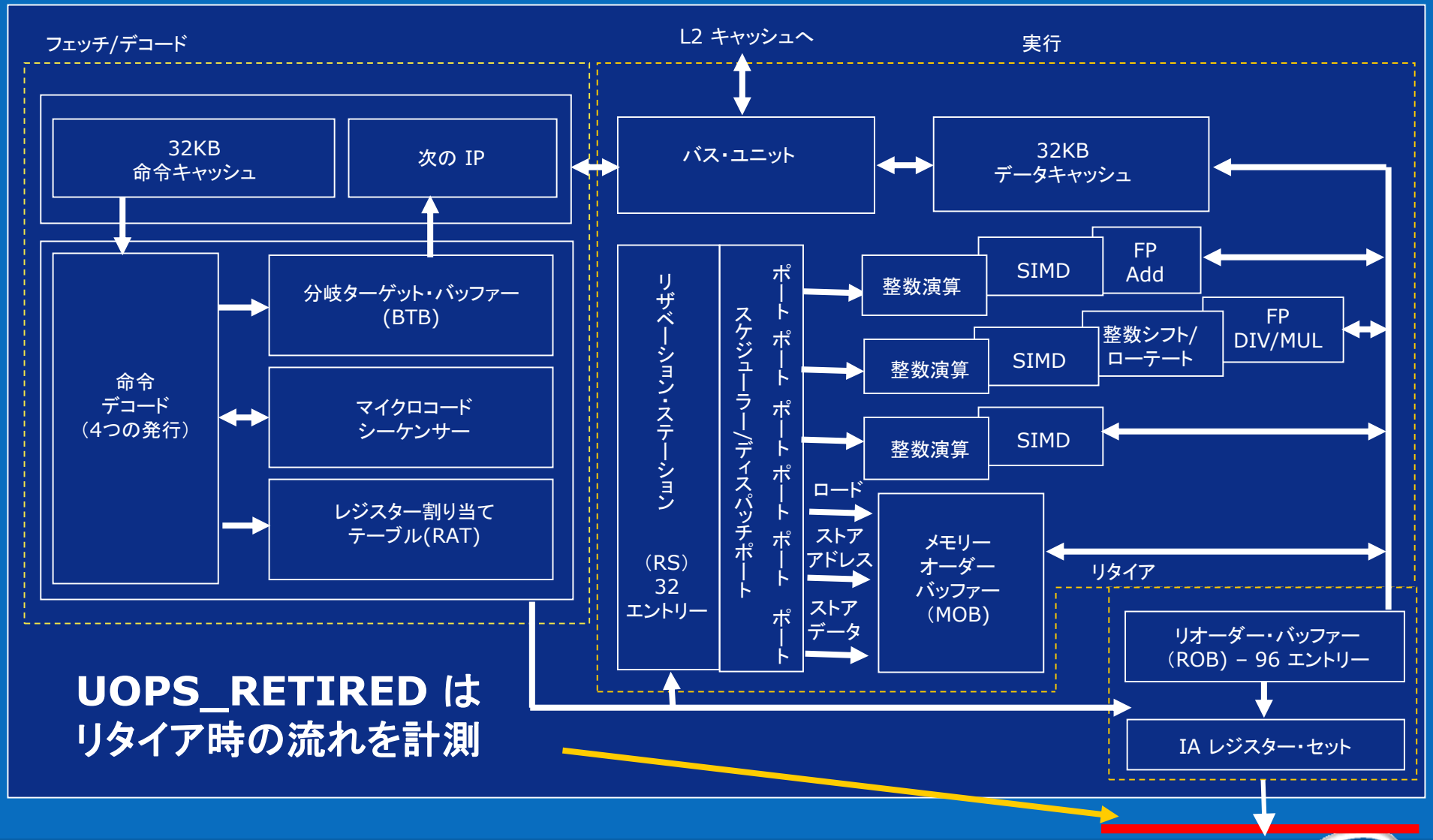


RESOURCE_STALL の要素

OOO エンジンへの UOP の流れは下流の原因でブロックされる

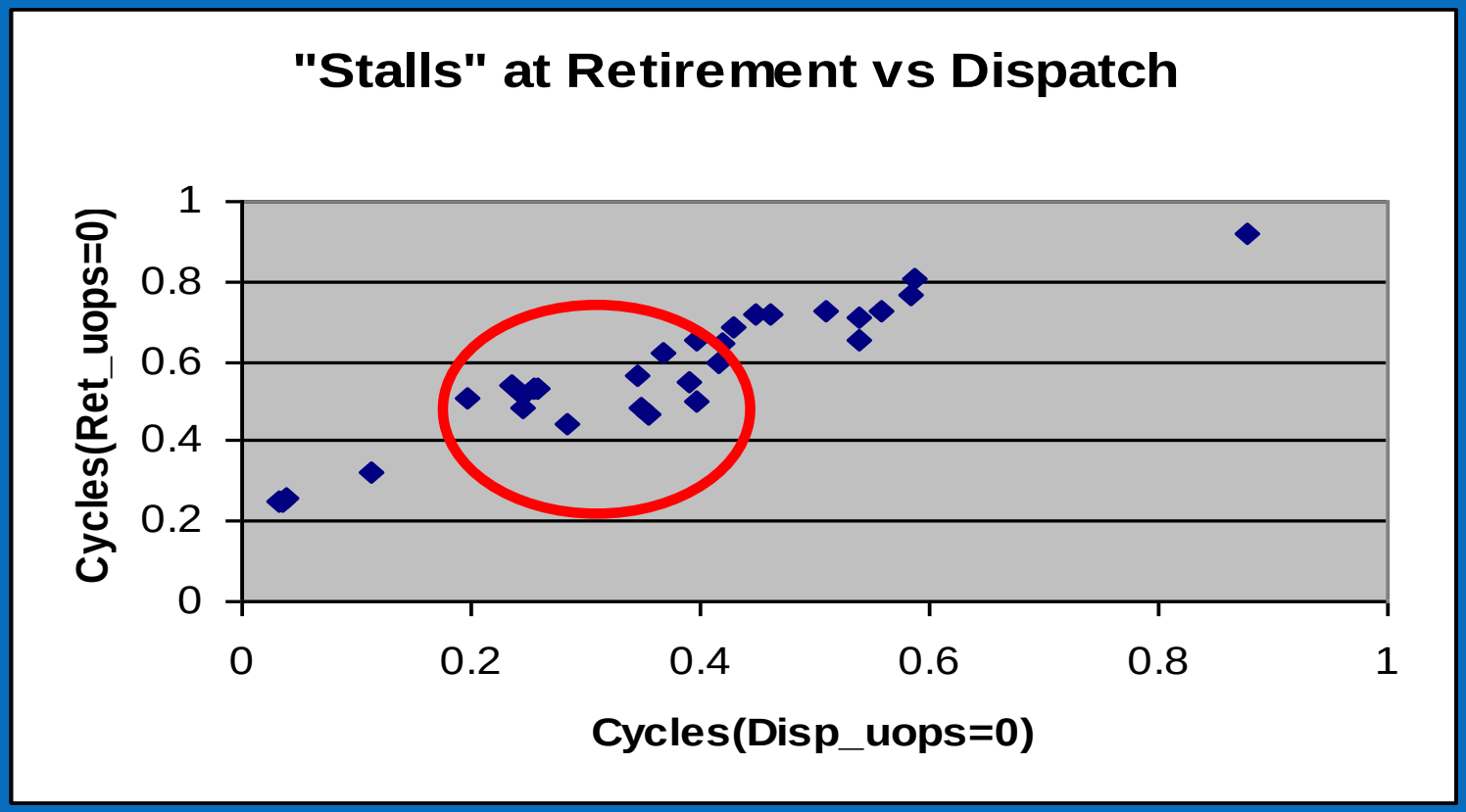
RESOURCE_STALLS.BR_MISS_CLEAR
　分岐予測ミスによるパイプライン・フラッシュによるストール
RESOURCE_STALLS.ROB_FULL
　ROB 中に 96 個の命令がある
RESOURCE_STALLS.LD_ST
　すべてのストアもしくはロード・バッファーが使用中
RESOURCE_STALLS.RS_FULL
　RS で 32 個の命令が入力待ちの状態
RESOURCE_STALLS.FPCW
　浮動小数点制御ワード（FPCW）の書き込みストール

アーキテクチャー・ブロック・ダイアグラム



インテル® ソフトウェア開発製品

リタイアとディスパッチによるストール、どちらを解決するか？



ループ中の差は OOO 実行による影響
ストールがディスパッチで発生している場合、
わずかな誤検出がある

x86 におけるサイクル計算

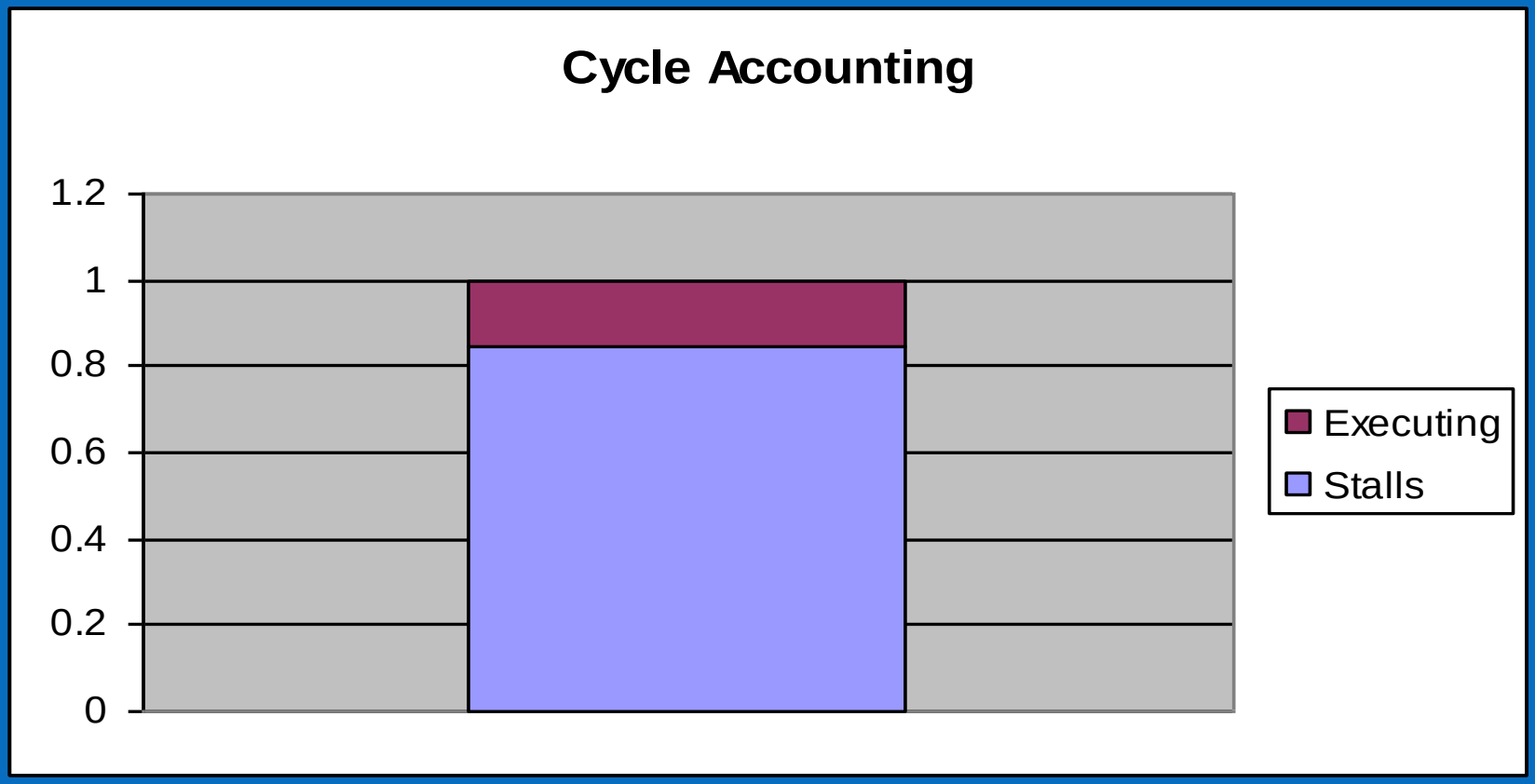
- サイクル ＝
  rs_uops_dispatched.cycles_none + rs_uops_dispatched.cycles_any
  “ストール” ＋ ディスパッチ

- サイクル ~ CPU_CLK_UNHALTED.CORE
  - CPU 依存の高いアプリケーション向け

x86 におけるサイクル計算

- ディスパッチ ~ cycles_dispatch_retiring_uops + cycles_dispatch_non_retiring_uops
  - リタイア/非リタイア UOP のオーバーラップはないと仮定
- 非リタイア UOP = rs_uops_dispatched – (uops_retired.any + Uops_retired.fused)
  - 非リタイア UOP サイクル ~ リタイアしていない uops / avg_uops_per_cycle
- 断片的に浪費された時間 = rs_uops_dispatched / (uops_retired.any + uops_retired.fused) - 1

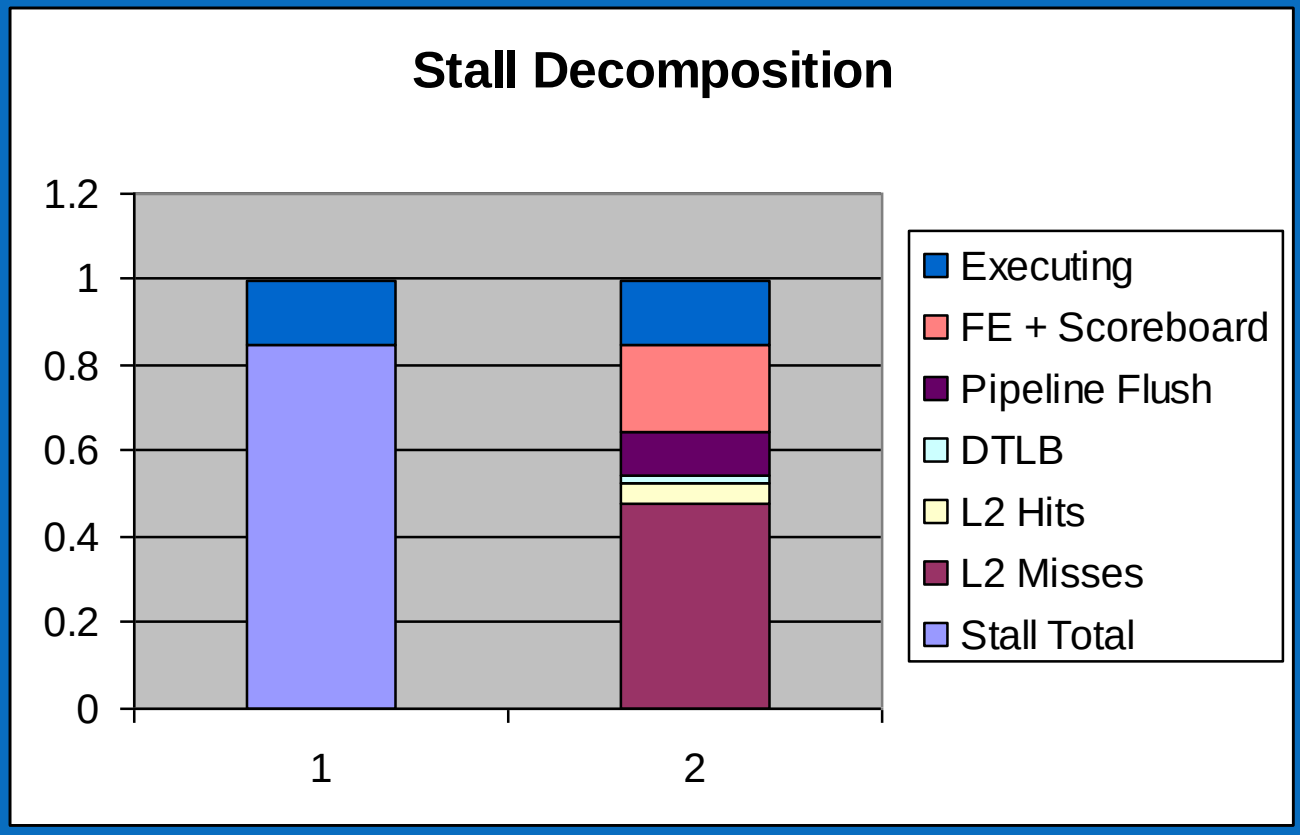
サイクル計算をまとめる



ストール・サイクル = UOP がディスパッチされていないサイクル
= RS_UOPS_DISPATCH.CYCLES_NONE

グラフは例題として作成したもので、実際の集計データではありません。

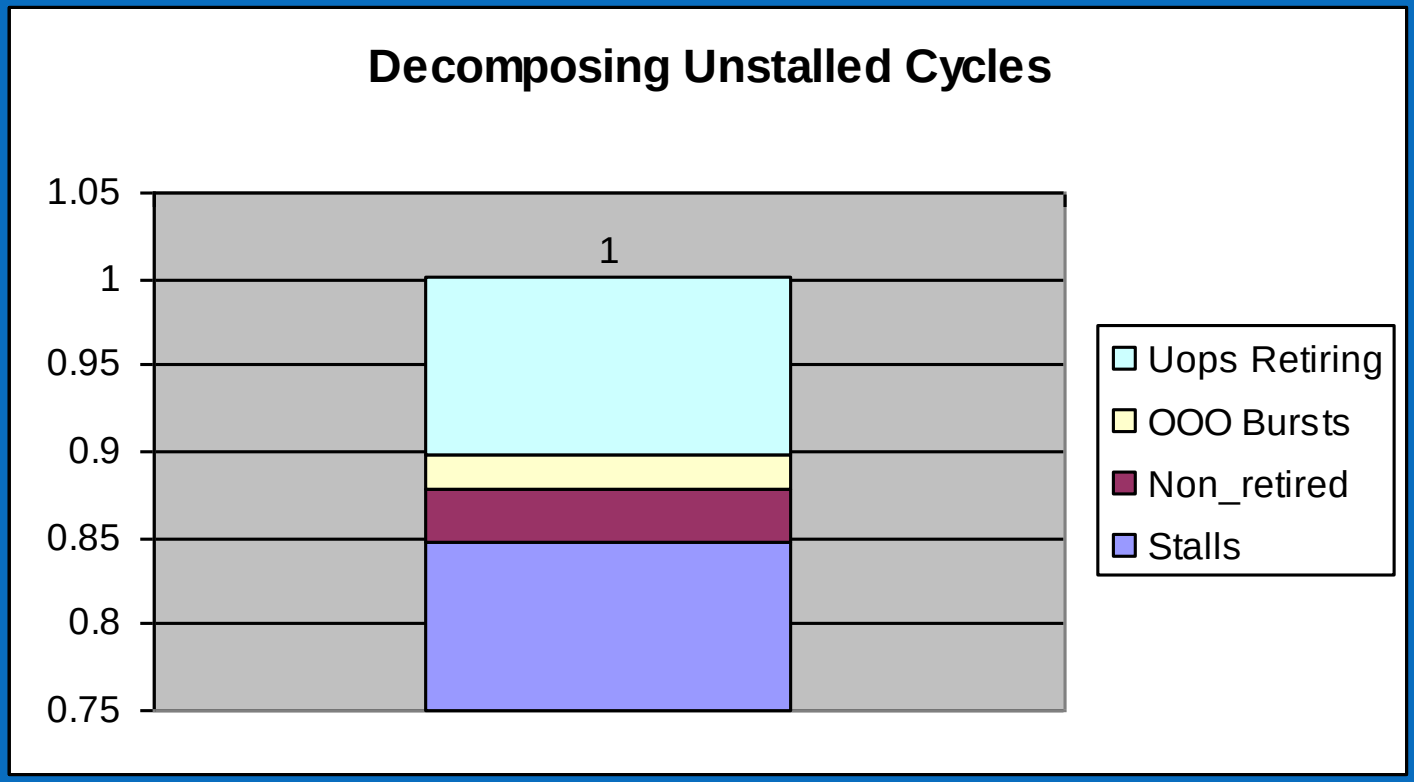
ストール・サイクルの分類



パイプライン・フラッシュ = RESOURCE_STALLS.BR_MISS_CLEAR / サイクル
L2 ヒット= (MEM_LOAD_RETIRED.L1D_LINE_MISS - MEM_LOAD_RETIRED.L2_LINE_MISS)*12/サイクル
DTLB/L2 ミス = イベント・カウント * ペナルティー / サイクル
FE + スコアボード = ストール – 上記の全て

グラフは例題として作成したもので、実際の集計データではありません。

ストールしていないサイクルを分類

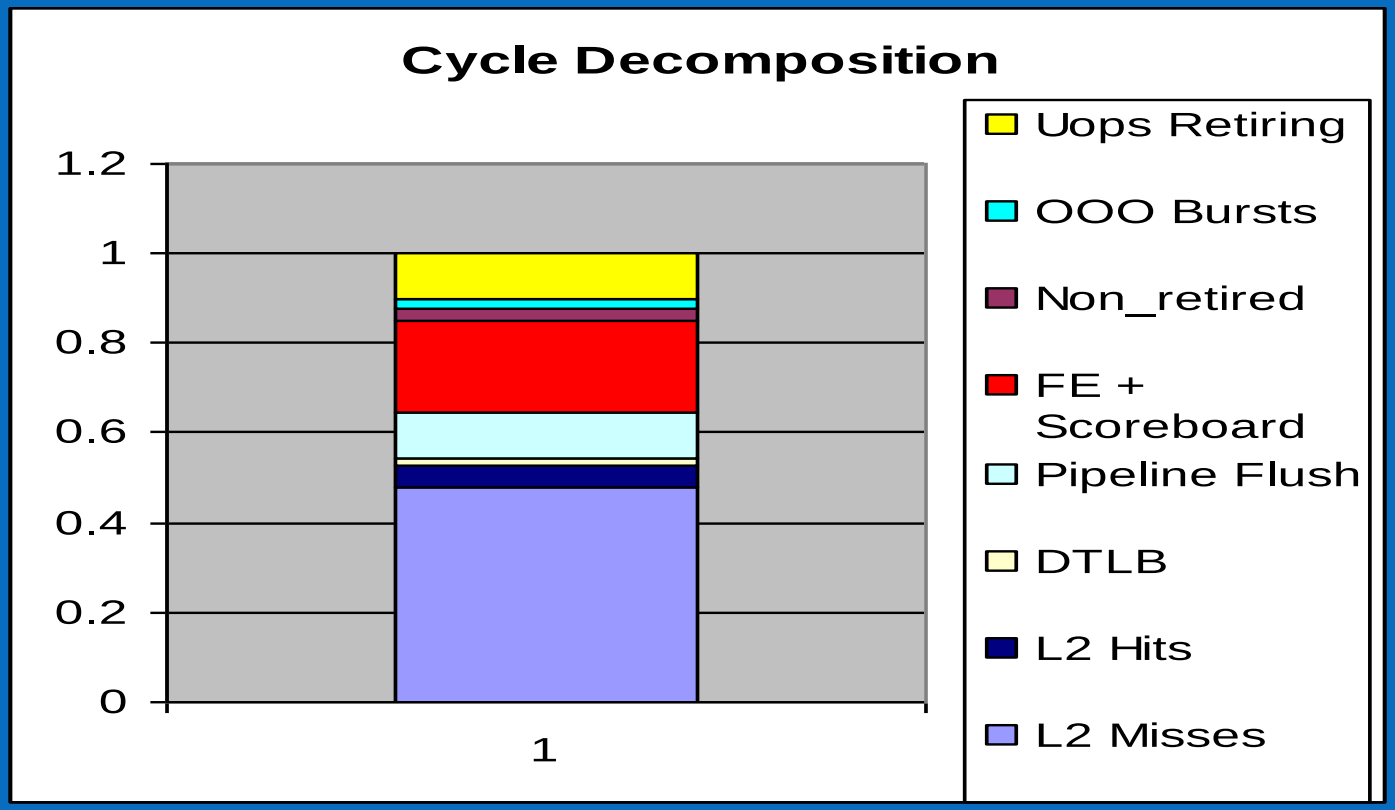


非リタイア =
((1 – (Uops_retired.any+Uops_retired.fused)/RS_Uops_Dispatched) * RS_Uops_Dispatched:cmask=1 / CPU_CLK_UNHALTED.CORE

OOO バースト = Uops_Retired.Any.cycles_none ー ストール – 非リタイア

グラフは例題として作成したもので、実際の集計データではありません。

全てのサイクルを分類



オーバー・カウントのリスク / FE の最少化 + スコアボード

しかし、非効率な命令実行解決へのガイドとなる

グラフは例題として作成したもので、実際の集計データではありません。

アーキテクチャー上の問題:

問題	カウンター	ペナルティー
先行する不明なアドレスからのロード位置へのストア	Load_Blocks.ADR	~5
8 バイトの中間から 4 バイトをストアフォワードする	Load_Blocks.Overlap_Store	~6
先行する N*4096 のオフセットをもつ不明なアドレスからのロード位置へのストア	Load_Blocks. Overlap_Store	~6
L2 からの 2 キャッシュ・ラインのロード	Load_Blocks.UNTIL_RETIRE	~22
先行するストアを伴う L2 からの 2 キャッシュ・ラインのロード	Load_Blocks.UNTIL_RETIRE	~20
プリフィックスによる読み込みレングスの変更(16 ビット即値)	ILD_STALLS	~6

“FE + スコアボード” へ影響する

ストアフォワード違反の検出

LOAD_BLOCK.STD（不明なデータサイズのストアによってロードがブロックされる）イベントは、アプリケーションでストアフォワード違反が発生したことを示す

ストア・フォワーディングとは?

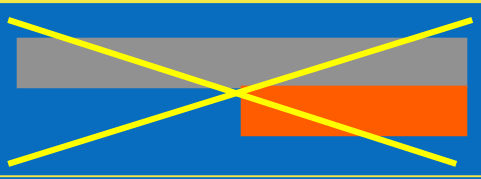
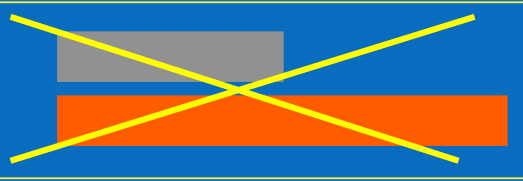
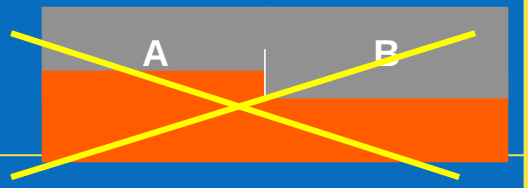
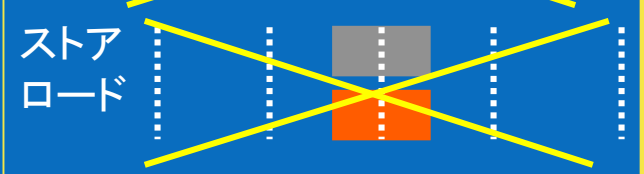
ストア・フォワーディングとは、ストアがメモリー階層に確定される前に、ストアのターゲットであったアドレスからロードしてストアを完了すること

ストアフォワード違反は、ロードとストアが適切にアライメントされていない場合やサイズに問題がある場合に発生する

- 以前のコンパイラーでアセンブリーを手動で記述した場合にのみ問題となる

プロセッサーのボトルネック

ストアフォワード違反

シナリオ	フォワードされる（違反なし）	フォワードされない（違反あり）
大きなストアの後の小さなロード		
同じサイズのロードとストア		
同じサイズのロードとストア		
128 ビット・フォワードは 16 バイトでアライン		

ストアフォワード違反

小さなデータをストアした後の大きなデータロード

```
mov DWORD PTR [esp+10h], 00000000h      // DWORD を esp+10h へストア
mov BYTE PTR [esp+10h], bl              // BYTE を esp+10h へストア
mov eax, DWORD PTR [esp+10h]            // ロードストール
and eax, 0xff
```

小さなストアの後の浮動小数点ロード

```
mov mem, eax            // DWORD を mem へストア
mov mem + 4, ebx        // DWORD を mem+4 へストア
fld   mem               // mem からの QWORD がストール
```

演習: ストアフォワード違反の検出

インテル® VTune™ パフォーマンス・アナライザーを使用して、ストアフォワード違反を検出する

演習テキスト:インテル® Core™ マイクロアーキテクチャーのパフォーマンス・カウンター
演習 2: ストアー・フォワード違反の検出を参照

分割されるロードとストア

キャッシュライン境界にまたがるロード/ストアが原因

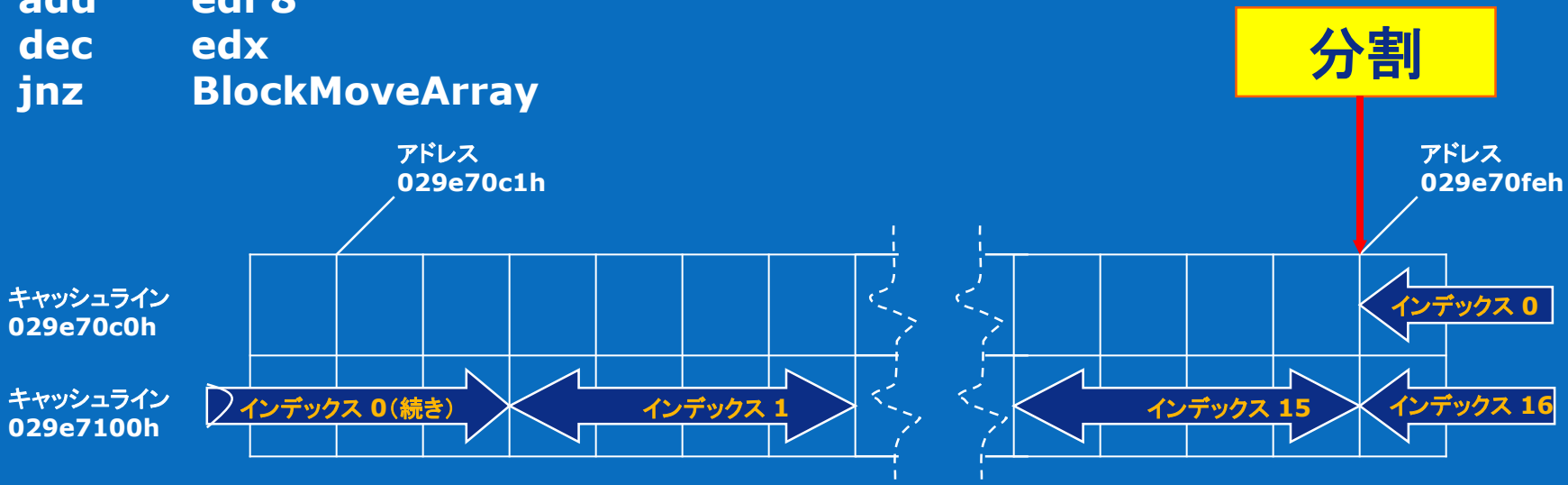
分割されることにより追加のメモリーアクセスが発生

次のイベントを監視することで測定可能

- Cache Line Load Split Reference Rate（分割ロードの数）
- Cache Line Store Split Reference Rate（分割ストアの数）

キャッシュライン分割の例

```
mov     esi, 029e70feh
mov     edi, 05be5260h
BlockMoveArray:
mov     eax, DWORD PTR [esi]
mov     ebx, DWORD PTR [esi+4]
mov     DWORD PTR [edi], eas
mov     DWORD PTR [edi+4], ebx
add     esi, 8
add     edi 8
dec     edx
jnz     BlockMoveArray
```



マルチコアのボトルネック

データの False Sharing

異なるスレッドによる同じキャッシュラインのロード/ストアが原因

パフォーマンスに与える影響が大きい

MEM_LOAD_RETIRED.L2_LINE_MISSイベントで測定する

適切なデータ構造体を 128 バイトにパディングするか、
またはアルゴリズムをコーディングし直して解決する

演習： False Sharing 問題の検出

インテル® VTune™ パフォーマンス・アナライザーを使用して、False Sharing 問題を検出する

演習テキスト：インテル® Core™ マイクロアーキテクチャーのパフォーマンス・カウンター
演習 3: False Sharing 問題の検出を参照

より詳しいイベントの階層

第 1 レベルのイベント	SAV (Sample After Value)
CPU_CLK_UNHALTED.CORE	2,000,000
RS_UOPS_DISPATCHED.CYCLES_NONE	2,000,000
UOPS_RETIRED.ANY + UOPS_RETIRED.FUSED	2,000,000
RS_UOPS_DISPATCHED	2,000,000
MEM_LOAD_RETIRED.L2_LINE_MISS	10,000
INST_RETIRED.ANY_P	2,000,000
ループ	
BUS_TRANS_ANY.SELF	100,000
BUS_TRANS_ANY.ALL_AGENTS	100,000
分岐	
RESOURCE_STALLS.BR_MISS_CLEAR	2,000,000

SAV 値はサンプルの比率となり、ペナルティーを吸収する

イベントの階層

第 2 レベルのイベント	SAV (Sample After Value)
MEM_LOAD_RETIRED.DTLB_MISS	20,000
MEM_LOAD_RETIRED.L2_MISS	10,000
MEM_LOAD_RETIRED.L1_LINE_MISS	200,000
BR_CND_MISSP_EXEC	2,000,000
BR_CND_EXEC	2,000,000
BR_CALL_EXEC	200,000
BR_CALL_MISSP_EXEC	200,000
ILD_STALLS	200,000
LOAD_BLOCK.STORE_OVERLAP	200,000

SAV 値はサンプルの比率となり、ペナルティーを吸収する

例: L1 ミス / L2_hit のペナルティーは 10 サイクル

ループにおける方法論？

- ホットな関数を探し最適化を行う
  - アライメントの解決、ベクトル化を促進するためループを分割
- バスのバンド幅に制限される関数を特定
  - FP 命令に制限されるループとマージする
- L2 キャッシュ・ミスを特定しソフトウェア・プリフェッチを行う
- OOO エンジンの流れを最適化する
  - ループの分割が有効

まとめ

インテル® Core™ 2 プロセッサーはループを効率よく実行する
- 2 つの SIMD 命令を 1 サイクルで実行
- みじかなパイプライン

インテル® Core™ 2 プロセッサーの PMU は優れている
- バンド幅を簡単にそして明確に定義可能
  - アドレス・バスの利用率は無いと仮定
- より多くのプリサイス・イベント
  - 命令リタイアとそのコンポーネント
  - LLC ミス
- ループ中の非効率な命令実行は予測可能

参考資料

インテル® VTune™ パフォーマンス・アナライザーを使用すると、ソフトウェアに含まれているマイクロアーキテクチャー上のボトルネックを検出できる

インテル® Xeon® プロセッサーおよびインテル® Pentium® 4 プロセッサー上でのソフトウェアの最適化に関する資料は、インテルの Web サイトを参照

インテルでは、以下のような情報を提供

- 日本語プロセッサー・テクニカル・ドキュメント
  - **http://www.intel.co.jp/jp/developer/download/**
- インテル® ソフトウェア開発製品
  - **http://www.intel.co.jp/jp/developer/software/products/**
- テクニカル情報、サポートしている環境、ホワイトペーパー
  - **http://developer.intel.com（英語）**
- 製品サポート
  - **http://support.intel.com（英語）**
- フォーラム
  - **http://softwareforums.intel.com/ids（英語）**

参考資料

インテル® Core™ アーキテクチャーにおけるプリサイス・イベント

BR_INST_RETIRED.MISPRED

INST_RETIRED.ANY_P

MEM_LOAD_RETIRED.DTLB_MISS

MEM_LOAD_RETIRED.L1D_LINE_MISS

MEM_LOAD_RETIRED.L1D_MISS

MEM_LOAD_RETIRED.L2_LINE_MISS

MEM_LOAD_RETIRED.L2_MISS

SIMD_INST_RETIRED.ANY

X87_OPS_RETIRED.ANY

これらのイベントはプリサイス・イベントと呼ばれ、イベントが発生した時の命令実行ポインター（EIP）に即座に関連付けられる。そのため、イベントの設定にイベント・スキッドは無い。

データを待つことはボトルネックか?

メモリー・バンド幅を知る

メモリーバンド幅 =
64 * Bus_Trans_Mem * 周波数 /
CPU_CLK_UNHALTED

メモリー・コンポーネントの比率が 20% 未満の場合、メモリーからデータをフェッチするための待機時間はパフォーマンス・ボトルネックにはならない

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。
製品に付属の売買契約書『Intel's Terms and conditions of Sales』に規定されている場合を除き、インテルはいかなる責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、商品性に関する保証、第三者の特許権、著作権、その他、知的所有権を侵害していないことへの保証を含む) に関しても一切責任を負わないものとします。

インテル製品は、予告なく仕様が変更されることがあります。

* 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。