

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



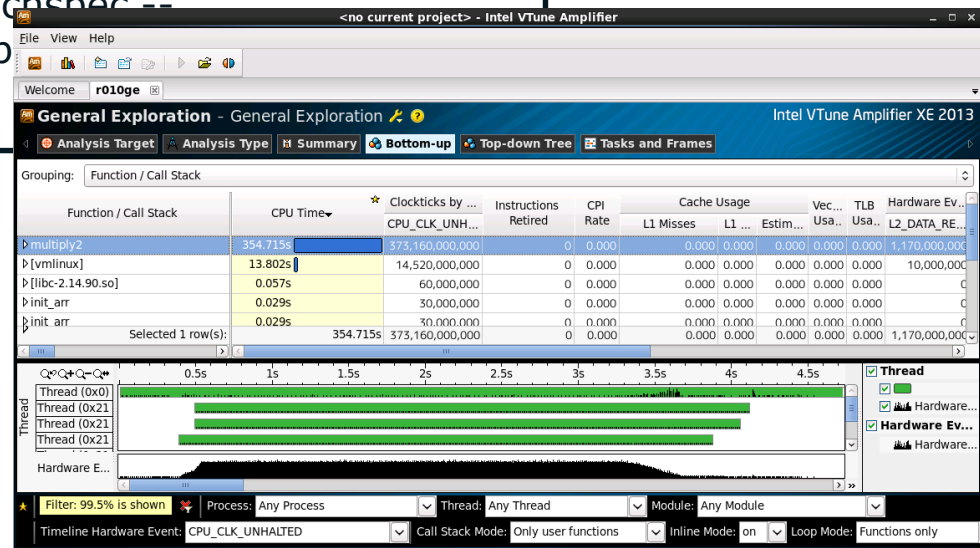
インテル® VTune™ Amplifier XE の概要

- ◆ 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

サンプリング手法

◆ 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、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
サポート非対象 CPU: インテル® Pentium® 4 プロセッサ・ファミリー以前、インテル以外のプロセッサ
- NMI Watchdog の設定は無効にする。

プリセットされた解析タイプ

Choose Analysis Type

Analysis Type

- Algorithm Analysis
 - Basic Hotspots
 - Advanced Hotspots
 - Concurrency
 - Locks and Waits
- Intel Core 2 Processor Analysis
- Nehalem / Westmere Analysis
- Sandy Bridge / Ivy Bridge / Haswell A
 - General Exploration
 - Bandwidth
 - Access Contention
 - Branch Analysis
 - Client Analysis
 - Core Port Saturation
 - Cycles and uOps
 - Memory Access
 - Port Saturation
- Intel Atom Processor Analysis
- Knights Corner Platform Analysis
 - Hotspots
 - General Exploration
 - Bandwidth
- Power Analysis
 - Custom Analysis

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)

解析タイプ ビューポイント(設定変更可能)

Concurrency Hotspots by Thread Concurrency

Intel VTune Amplifier XE 2013

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Top-down Tree | Tasks

Grouping: Function / Thread / Call Stack

Function / Thread / Call Stack	CPU Time by Utilization	Over... Time	Wait Time by Utilization
grid_intersect	4.146s	0s	
initialize_2D_buffer	3.438s	0s	
sphere_intersect	2.750s	0s	
thread_trace\$omp\$parallel@241	2.275s	0.147s	5.811s
NtDelayExecution	0.437s		
grid_bounds_intersect	0.415s		
kmp_end_split_barrier	0.412s		

Bottom-up ペイン
Top-down Tree ペイン

CPU Function/CPU Stack - CPU Time
Viewing 1 of 24 selected stack(s)
36.7% (1.523s of 4.146s)

build_serial.exe!grid_intersect - grid.cpp
build_serial.exe!shader+0x39f - shade...
build_serial.exe!trace+0x91...
build_serial.exe!render_one_pixel+0x91...
build_serial.exe!thread_trace\$omp\$par

コール・スタック・ペイン

タイムライン・ペイン

Thread: WinMainCRTStartup (0x1e08), DLLUnregisterServer (0x1ab8), Thread (0x1ebc), GdiipCreateSolidFill (0x1ec8), Thread (0x1ec8), CoGetTreatAsClass (0x1ed4), Thread (0x1ed4), thread_video (0x1edc), __kmp_launch_monitor (0x1ee4), OMP Worker Thread #1 (0x1eec)

CPU Usage
Thread Concurrency

Thread: Running, Waits, CPU Time, User Tasks, Transitions
CPU Usage: CPU Time
Thread Concurr...: Concurrency

No filters are applied. Any Process | Any Thread | Any Module | Utilization: Any Utilization
Call Stack Mode: Only user functions | Inline Mode: on

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・レベルヘッドリルダウンして結果を表示

General Exploration - Hardware Issues Intel VTune Amplifier XE 2013

Analysis Target Analysis Type Collection Log Summary Bottom-up Top-down Tree Tasks grid.cpp

Source Assembly

So... Line	Source	CPU_CLK_U...	CPU_C...*	INST_RE...	Code Location	So... Line	Assembly	CPU_CLK...	CPU_C...*	INST_RE...	OFF..	O
570	tmax.x += tdelta.x;											
571	curpos = nXp;											
572	nXp.x += pdeltaX.x;											
573	nXp.y += pdeltaX.y;											
574	nXp.z += pdeltaX.z;	2,000,000	2,000,000	6,000,000	0x40e001	577	jbe 0x40e080 <Block 47>	428,000,0...	488,000,...	434,000,...	0	ε
575	}						Block 38:					
576	else if (tmax.z < tmax.y) {	46,000,000	38,000,000	20,000,000	0x40e003	578	test edi, edi	2,000,000	0	6,000,000	0	
577	cur = g->cells[voxindex];	472,000,000	548,000,...	476,000,...	0x40e005	578	jz 0x40e04a <Block 44>	6,000,000	12,000,000	16,000,000	0	
578	while (cur != NULL) {	18,000,000	20,000,000	44,000,000			Block 39:					
579	if (ry->mbox[cur->obj->id] != ry->	6,038,000,000	6,216,00...	5,590,00...	0x40e007	579	mov edx, dword ptr [edi+0x4]	486,000,0...	434,000,...	456,000,...	0	
580	ry->mbox[cur->obj->id] = ry->sez	1,506,000,000	1,468,00...	1,116,00...	0x40e00a	579	mov eax, dword ptr [esi+0x10]	840,000,0...	848,000,...	872,000,...	0	
581	cur->obj->methods->intersect(cur	1,842,000,000	1,768,00...	1,918,00...	0x40e00d	579	mov ecx, dword ptr [esi+0xc]	44,000,000	40,000,000	36,000,000	0	
582	}				0x40e010	579	mov edx, dword ptr [edx]	104,000,0...	102,000,...	90,000,000	0	
583	cur = cur->next;	1,006,000,000	1,072,00...	990,000,...	0x40e012	579	cmp dword ptr [eax+edx*4], ecx	2,510,000...	2,640,00...	2,352,00...	0	
584	}				0x40e015	579	jz 0x40e040 <Block 42>	2,054,000...	2,152,00...	1,784,00...	0	
585	curvox.z += step.z;	84,000,000	88,000,000	80,000,000			Block 40:					
586	if (ry->maxdist < tmax.z curvox.z	160,000,000	148,000,...	168,000,...	0x40e017	580	mov edx, dword ptr [edi+0x4]	1,018,000...	960,000,...	816,000,...	0	
587	break;											
588												
589												
590												

Selected 1 row(s): 6,038,000,000 6,216,00... 5,590,00...

選択したソースコード・ラインに対応したアセンブリー・コードがハイライトされる

ソースレベルで収集データを表示

アセンブリー・レベルで収集データを表示 (ブロック単位での表示)

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

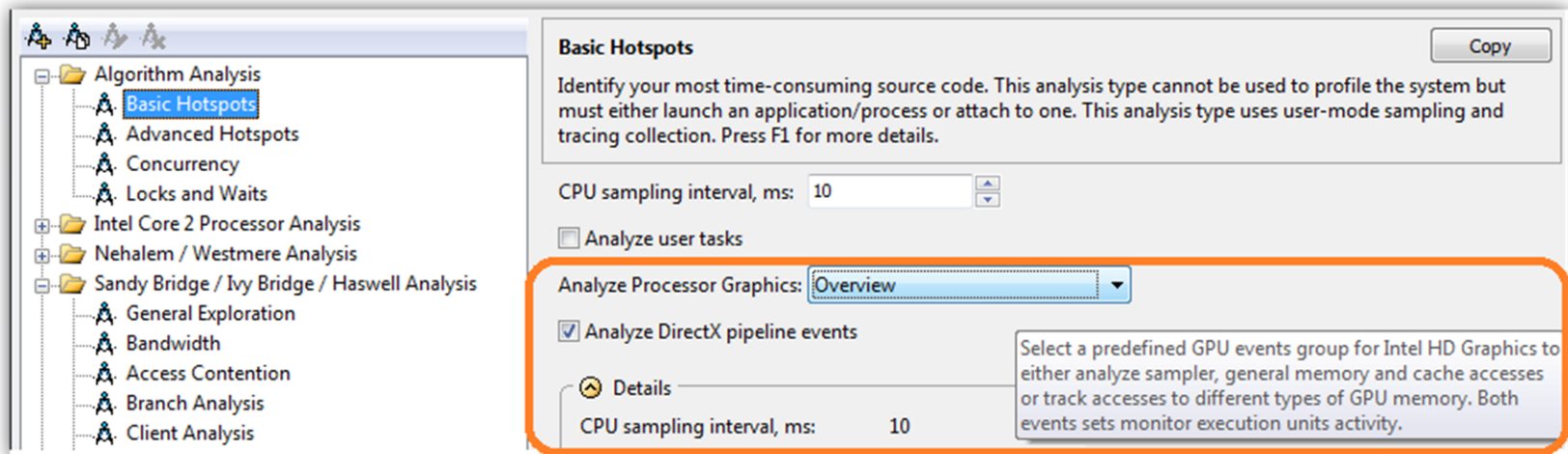
Uncore Event summary
-----
Hardware Event Type      Hardware Event Count:Self
-----
UNC_IMC_GT_REQUESTS     31635126
UNC_IMC_IA_REQUESTS     124751817
UNC_IMC_IO_REQUESTS     36705657
UNC_IMC_DATA_READS      148408289
UNC_IMC_DATA_WRITES     45161425
```

```
$ 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



The screenshot shows the Intel VTune Performance Analyzer interface. On the left is a tree view of analysis categories, with 'Basic Hotspots' selected under 'Algorithm Analysis'. The main panel is titled 'Basic Hotspots' and contains the following configuration options:

- Basic Hotspots** (Title bar with a 'Copy' button)
- Description: Identify your most time-consuming source code. This analysis type cannot be used to profile the system but must either launch an application/process or attach to one. This analysis type uses user-mode sampling and tracing collection. Press F1 for more details.
- CPU sampling interval, ms: 10
- Analyze user tasks
- Analyze Processor Graphics: Overview (dropdown menu)
- Analyze DirectX pipeline events
- Details section: CPU sampling interval, ms: 10

A callout box on the right side of the 'Analyze DirectX pipeline events' option provides additional information: "Select a predefined GPU events group for Intel HD Graphics to either analyze sampler, general memory and cache accesses or track accesses to different types of GPU memory. Both events sets monitor execution units activity."

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

Hotspots - Hotspots Intel VTune Amplifier XE 2013

Analysis Target Analysis Type Collection Log Summary Bottom-up Caller/callee Top-down Tree Tasks and Frames

Function	CPU Time:Total	CPU Time	Module
thread_video	9.126s	0s	analyze_locks.exe
tachyon_video::on_process	9.126s	0s	analyze_locks.exe
rt_renderscene	9.126s	0s	analyze_locks.exe
renderscene	9.126s	0s	analyze_locks.exe
trace_region	9.126s	0s	analyze_locks.exe
trace_shm	9.126s	0s	analyze_locks.exe
thread_trace	9.126s	0s	analyze_locks.exe
[TBB parallel_for on class draw_task]	9.126s	0s	analyze_locks.exe
tbb::interface6::internal::partition_type_base<class	9.126s	0s	analyze_locks.exe
draw_task::operator()	9.126s	0s	analyze_locks.exe
render_one_pixel	8.990s	0.010s	analyze_locks.exe
trace	8.960s	0s	analyze_locks.exe
intersect_objects	8.554s	0.020s	analyze_locks.exe
grid_intersect	8.514s	4.374s	analyze_locks.exe
shader	7.739s	0.223s	analyze_locks.exe
shade_reflection	4.134s	0.010s	analyze_locks.exe
sphere_intersect	3.432s	3.429s	analyze_locks.exe
_tmainCRTStartup	0.612s	0s	analyze_locks.exe
WinMain	0.612s	0s	analyze_locks.exe
main	0.612s	0s	analyze_locks.exe
video::main_loop	0.452s	0.230s	analyze_locks.exe
grid_bounds_intersect	0.370s	0.370s	analyze_locks.exe
InternalWndProc	0.332s	0.130s	analyze_locks.exe
loop_once	0.223s	0.021s	analyze_locks.exe
...	0.202s	0.202s	analyze_locks.exe
Selected 1 row(s):	8.554s	0.020s	

Callers	CPU Time:T...	CPU Time
intersect_objects	8.554s	0.020s
shader	5.500s	0.020s
trace	5.500s	0.020s
render_one_pixel	3.377s	0.020s
shade_reflection	2.124s	0s
shader	2.124s	0s
trace	3.053s	0s
shade_reflection	1.852s	0s
render_one_pixel	1.201s	0s

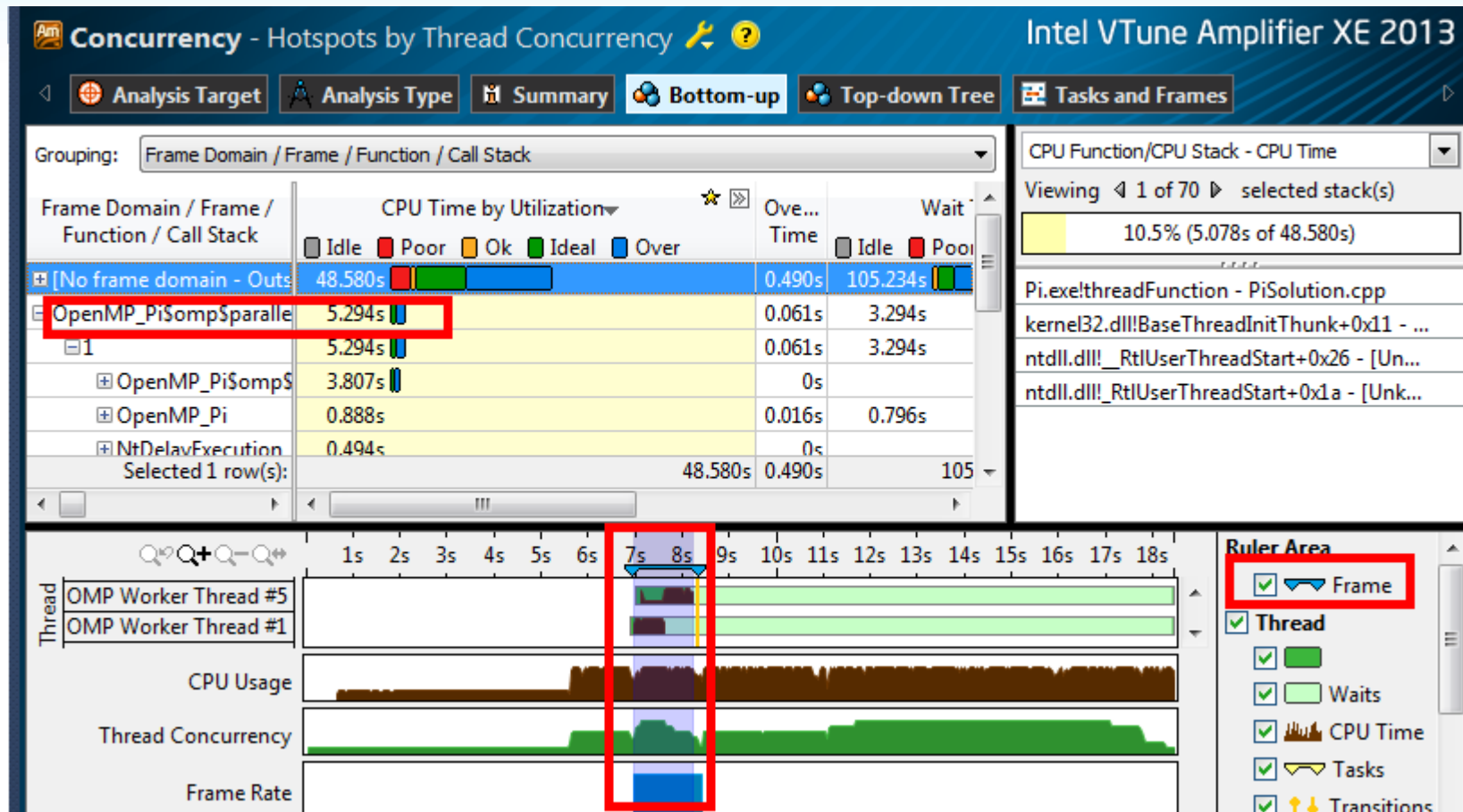
Callers	CPU Time:T...	CPU Time
intersect_objects	8.554s	0.020s
grid_intersect	8.514s	4.007s
sphere_intersect	3.331s	3.327s
add_intersection	0.004s	0.004s
grid_intersect	0.916s	0.367s
grid_bounds_intersect	0.086s	0.086s
tri_intersect	0.073s	0.062s
pos2grid	0.061s	0.061s
VScale	0.030s	0.030s
Raypnt	0.010s	0.010s
light_intersect	0.020s	0.020s

No filters are applied. Process: Any Process Thread: Any Thread Module: Any Module

Call Stack Mode: Only user functions Inline Mode: on Loop Mode: Functions only

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

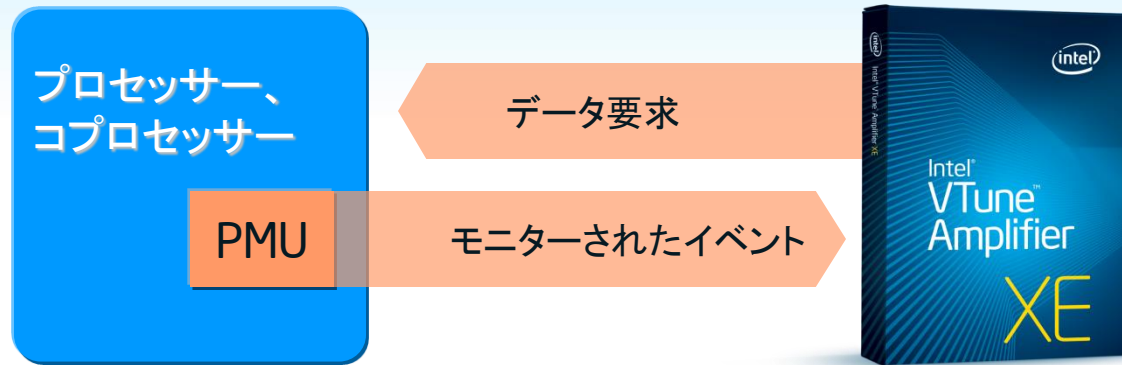
Function / Call Stack	CPU Time	Instructions Retired	CPI Rate	Module
OpenAdapter10	1.067s	2,274,621,910	1.819	igd10umd64.dll
draw	0.918s	1,892,099,285	1.880	[Dynamic code]
_updateMousePosition	0.699s	534,002,196	5.066	[Dynamic code]
AResample::resampleA411	0.453s	4,917,586,011	0.355	RESAMPLEDMO.DLL
draw	0.353s	638,293,807	2.145	[Dynamic code]
hybDriverEntry	0.284s	666,707,741	1.645	igdkmd64.sys
Recycler::ProcessMark	0.255s	987,146,359	0.984	jscript9.dll
[igd10umd64.dll]	0.224s	487,622,274	1.788	igd10umd64.dll
isVisible	0.138s	231,345,886	2.313	[Dynamic code]
clone	0.138s	367,544,372	1.465	[Dynamic code]
KelInvalidateRangeAllCaches	0.129s	203,474,509	2.370	ntoskml.exe

Hardware Event Sample
67.8% (0.622s of 0.918s)
[Dynamic code]!draw - BitmapSequence.js
[Dynamic code]!draw+0x34f0 - Container.js:115
[Dynamic code]!draw+0x34f0 - Container.js:115
[Dynamic code]!update+0x1aed5 - Stage.js:244
[Dynamic code]!tick+0x24de8 - Game.js:1458
[Dynamic code]!tick+0xb3b - Tick.js:73
jscript9.dll!amd64_CallFunction+0x7b - [Unknown]:[Unknown]
jscript9.dll!Js::JavascriptFunction::CallFunction+0x6b - [Unknown]:[Unk...
jscript9.dll!Js::JavascriptFunction::CallRootFunction+0x129 - [Unknown]...
jscript9.dll!ScriptSite::CallRootFunction+0x63 - [Unknown]:[Unknown]

JavaScript の解析結果

Code the Future

Performance Monitoring Unit (PMU)



- プロセッサ、コプロセッサに実装されている Performance Monitoring Unit (PMU) というバッファに、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

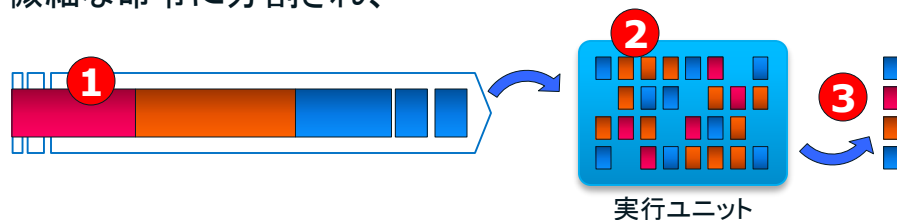
CPI Rate (Clockticks per Instructions Retired Rate)

$$\begin{aligned} \text{CPI Rate} &= \text{Clockticks} / \text{Instructions Retired} \\ &= \text{CPU_CLK_UNHALTED.THREAD} / \text{INST_RETIRED.ANY} \end{aligned}$$

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、

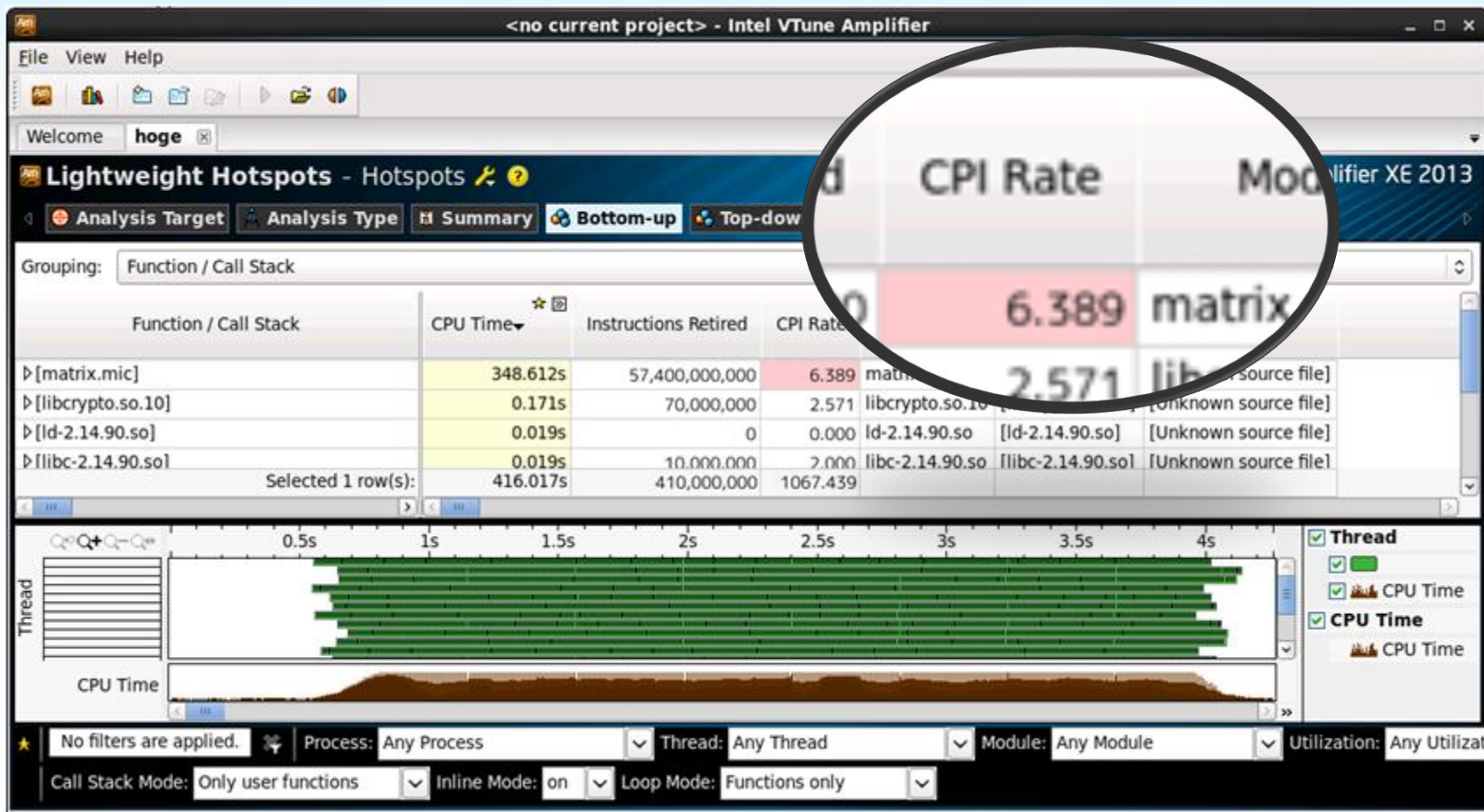


2. コア内では、実行可能なマイクロオペレーションから先に順次実行されます。

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™ コプロセッサの CPI

インテル® Xeon Phi™ コプロセッサ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® プロセッサは CPI=0.25 が最良であったが、
- ・インテル® Xeon Phi™ コプロセッサは CPI=2 が最良

インテル® Xeon Phi™ コプロセッサでは、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

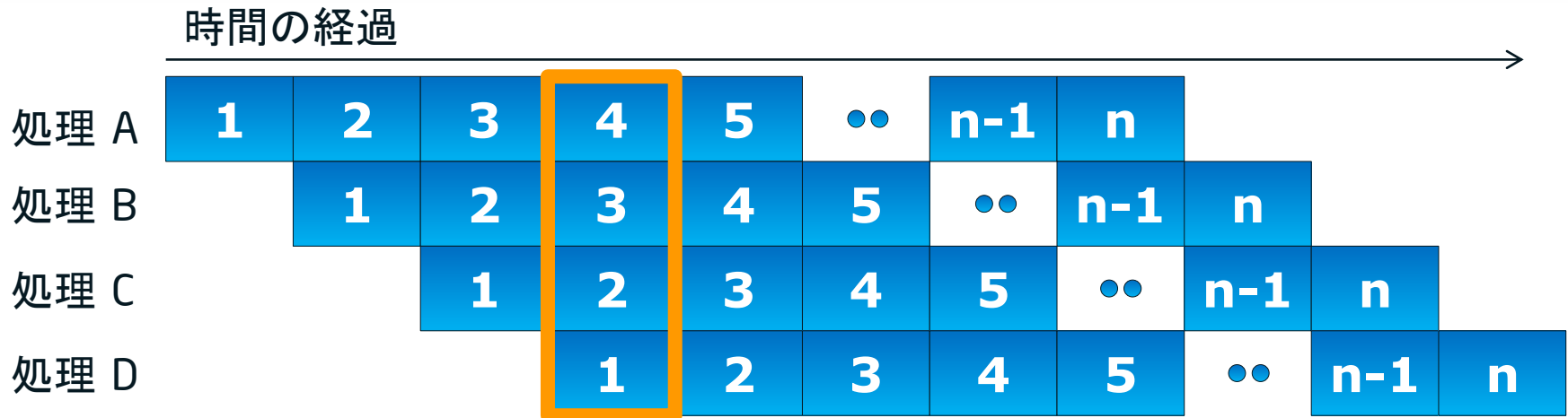
インテル® VTune™ Amplifier XE を使用した パフォーマンス分析の一般的な手順

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

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC	Nxt IP	TC	Fetch	Drive	Alloc	Rename	Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br	Ck	Drive

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

General Exploration - Hardware Issues   Intel VTune Amplifier XE 20

Analysis Target Analysis Type Collection Log Summary Bottom-up

/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	0.162
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;
}

```

```

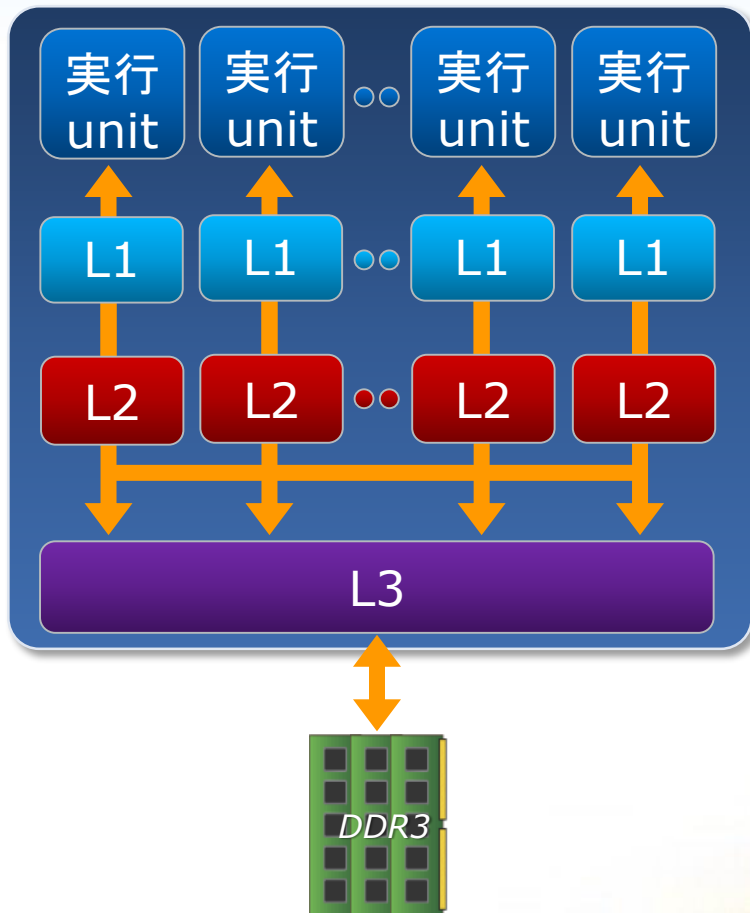
if ( blend = 255 )
    for ( j = 0; j < size; j++ )
        dest[j] = src_1[j];
else if ( blend == 0 )
    for ( j = 0; j < size; j++ )
        dest[j]= src_2[j];
else
    for ( j = 0; j < size; j++ )
        dest[j] = ( src_1[j] * blend + src_2[j] *
(255-blend) ) / 256;

```

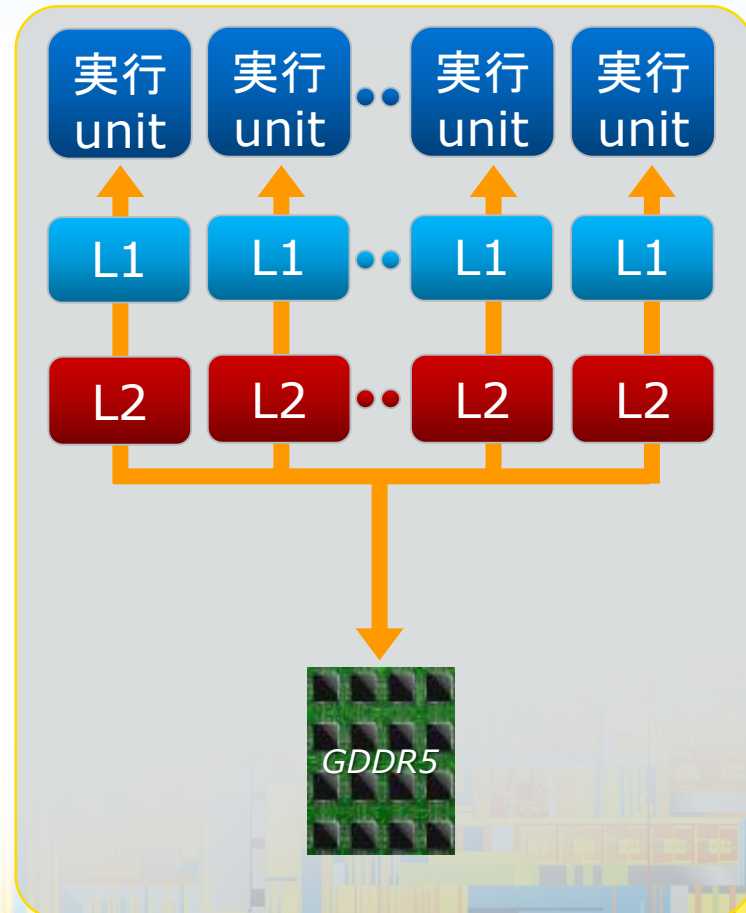
ループ内で評価不要な分岐をループの外に出して分岐予測ミスの発生をふせぐ

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

インテル® Xeon® プロセッサ

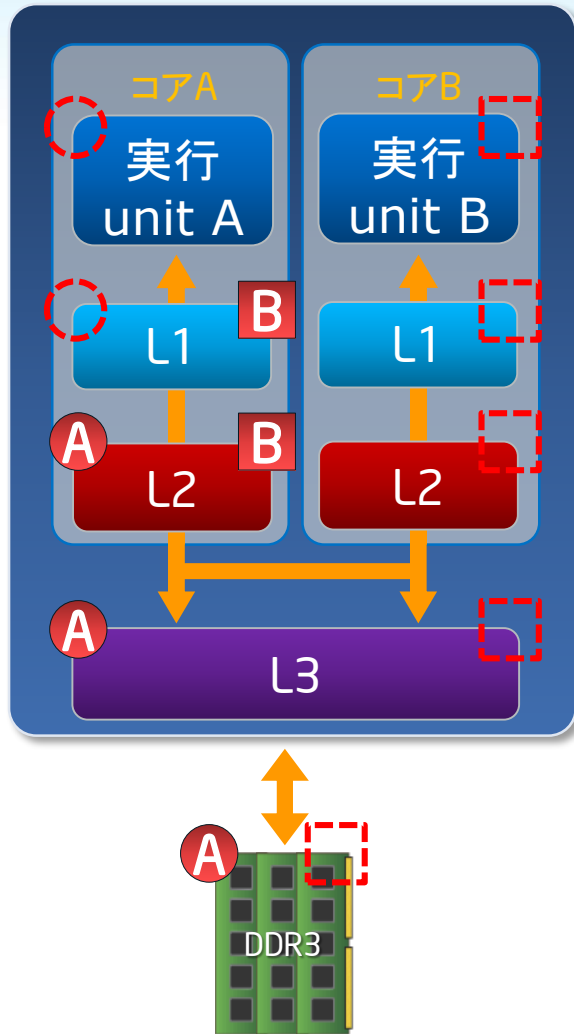


インテル® Xeon Phi™ コプロセッサ



Code the Future

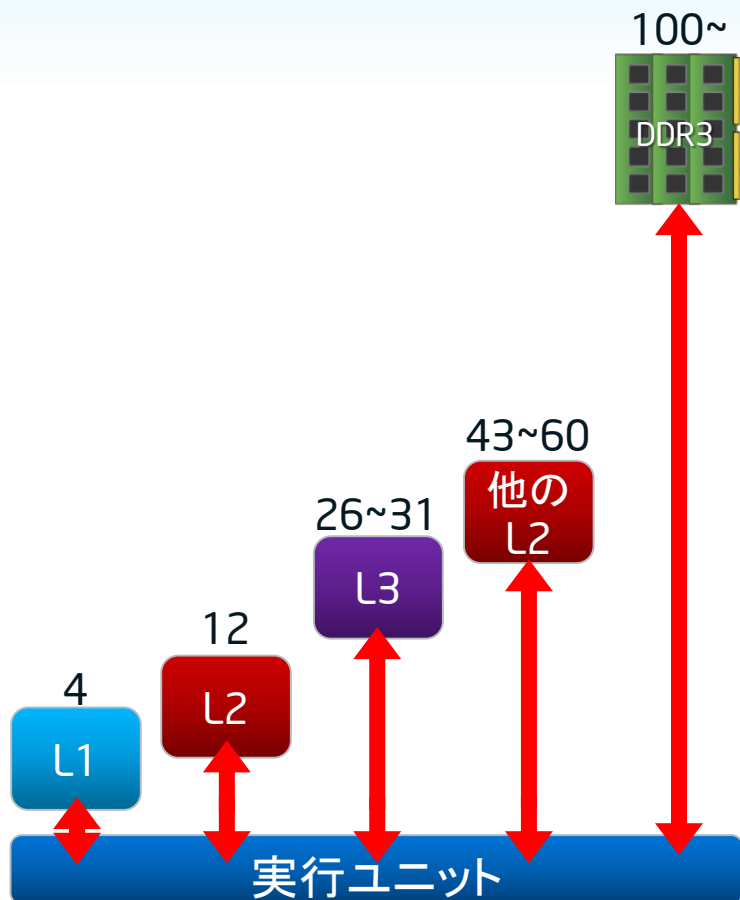
キャッシュによる高速化



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

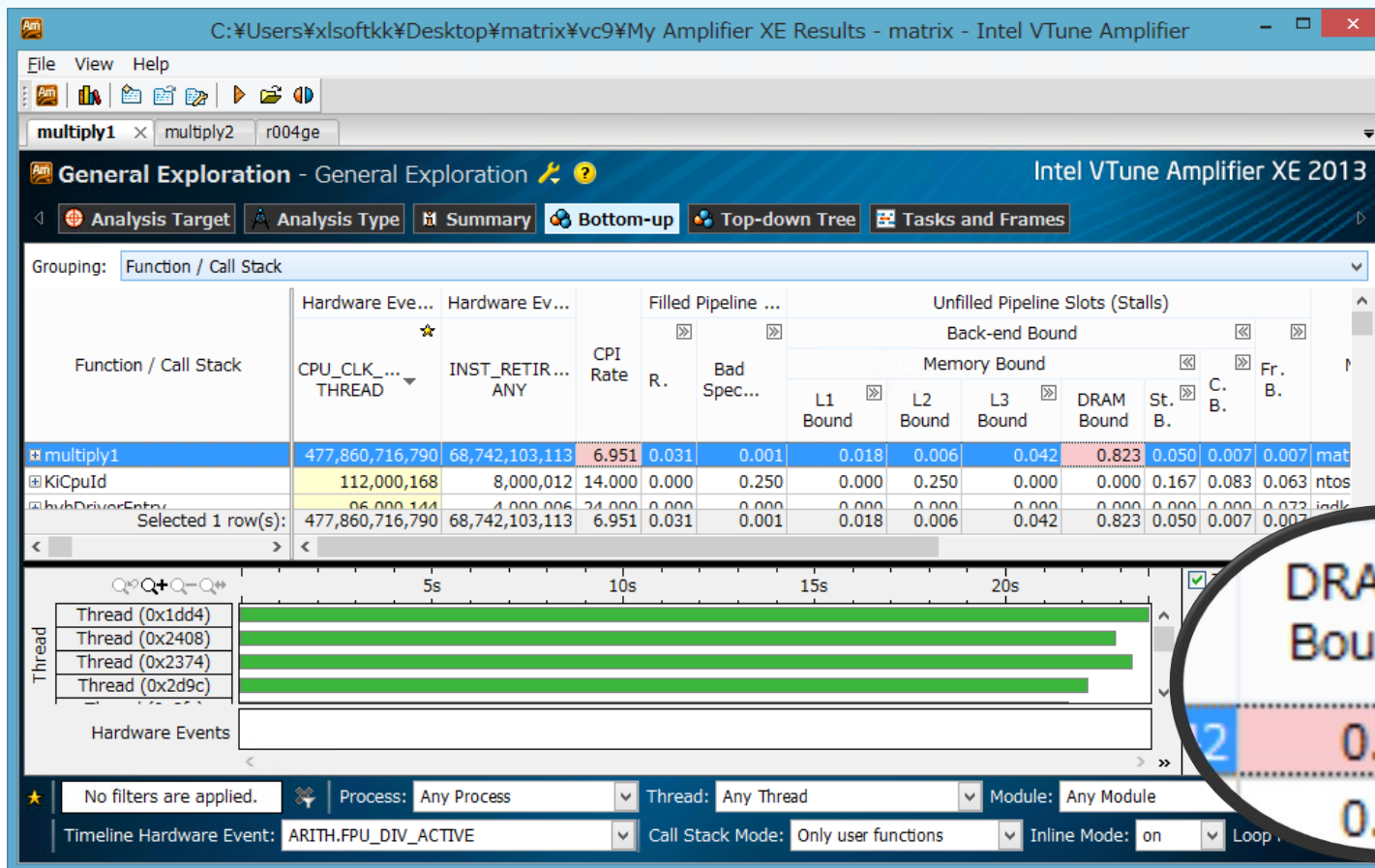


SandyBridge[†] アーキテクチャの転送時間 (単位は CPU サイクル)

	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
メモリ	~4TB	~16GB

キャッシュ、メモリーの容量

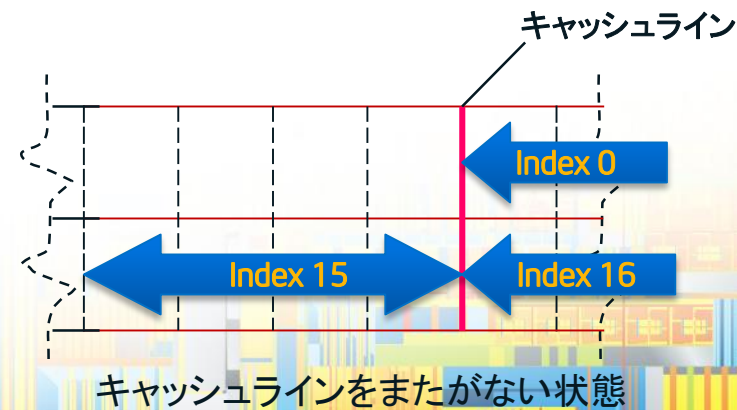
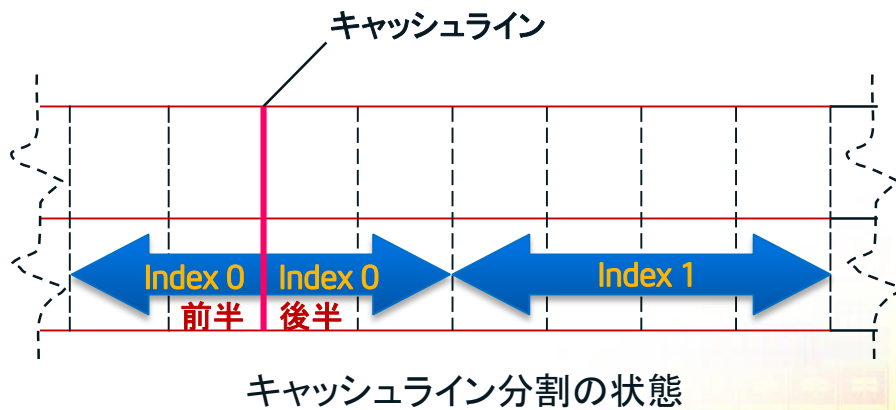
インテル® VTune™ Amplifier XE による データ転送によるパフォーマンス問題の分析



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



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)
```

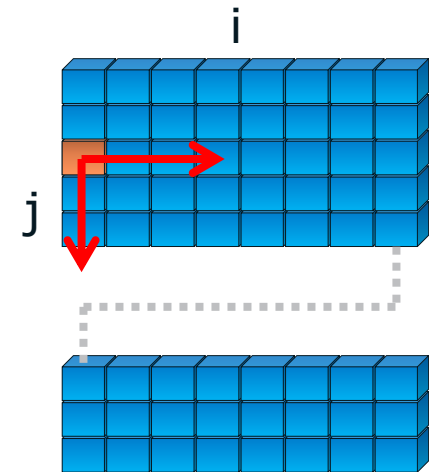
```
!DEC$ATTRIBUTES ALIGN: 64:: A
```

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][i] * C[j][i]
```

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



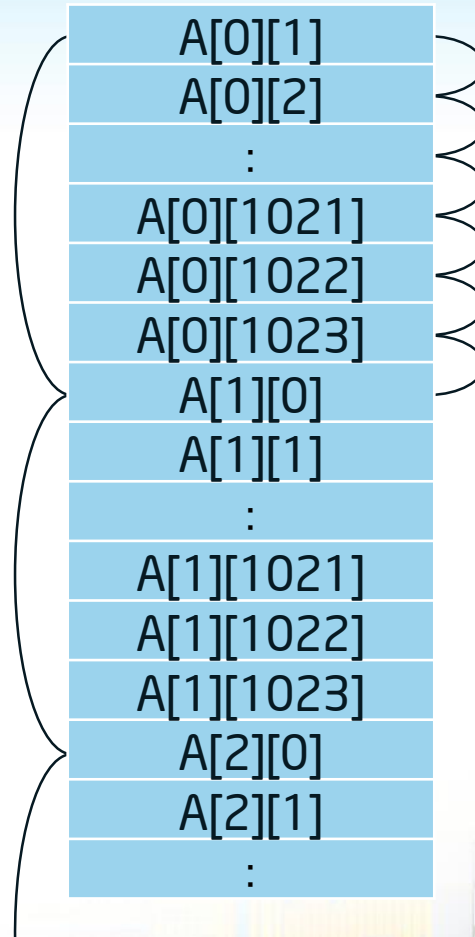
アクセスの効率性

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]
```

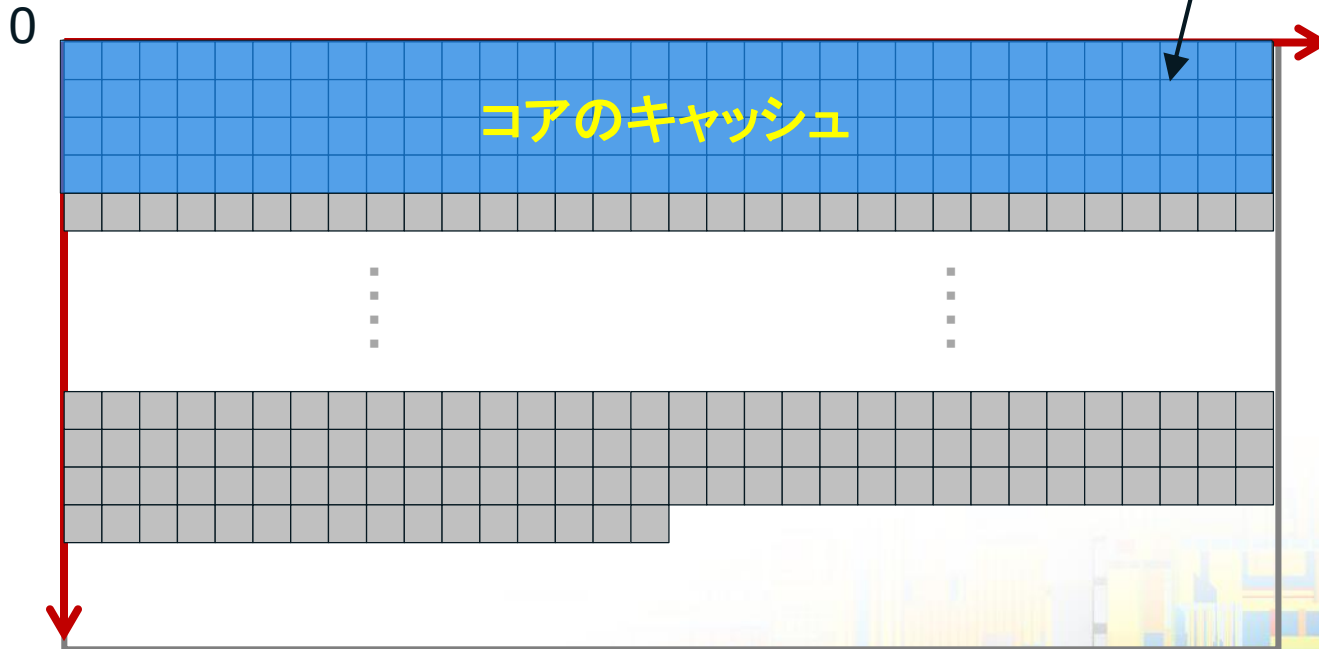


Code the Future

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];  
    }  
  }  
}
```

メモリー上の c[][] 領域



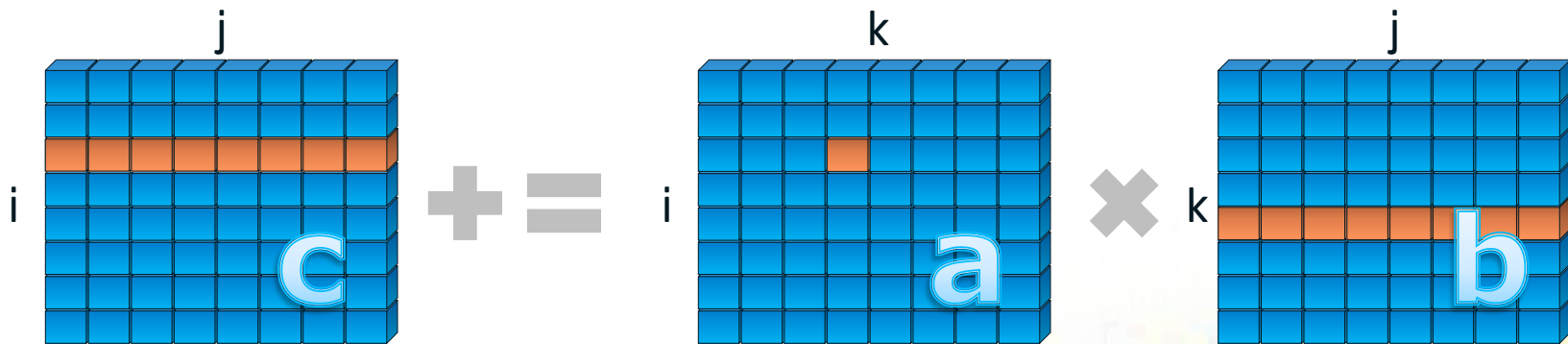
キャッシュにはすべてのデータが入らない

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];
    }
  }
}
```

```
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];
          }
        }
      }
    }
  }
}
```



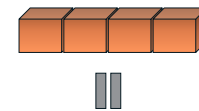
ブロッキング

Code the Future

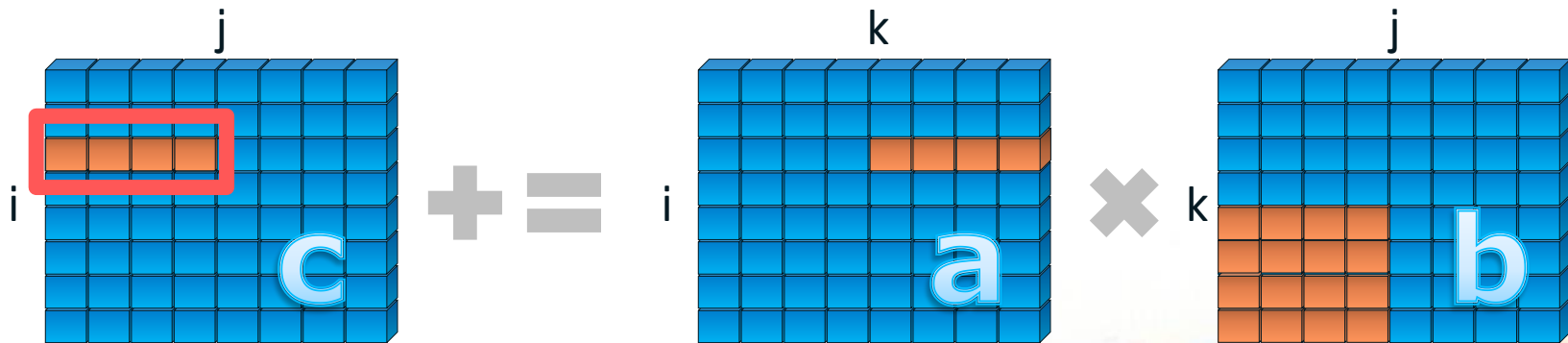
ブロッキングにより書き換えを防ぐ

```
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

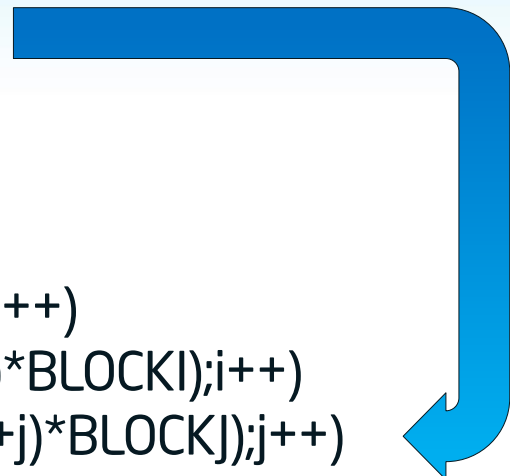



キャッシュ上のデータを書き換えないようにして (ブロッキング)
メモリの読み込みとキャッシュ入れ替えによる遅延の発生を防ぐ。

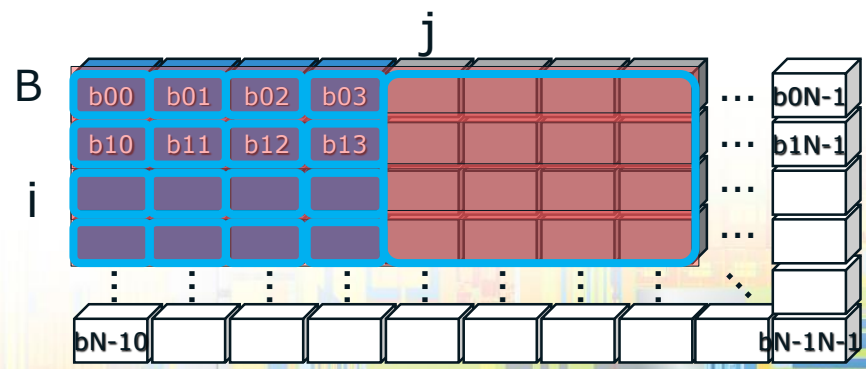
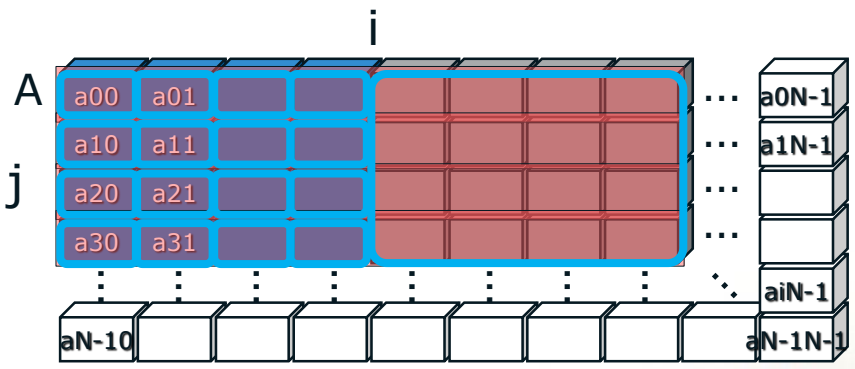
他のブロッキング例

```
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];
```



 = キャッシュライン・サイズ



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

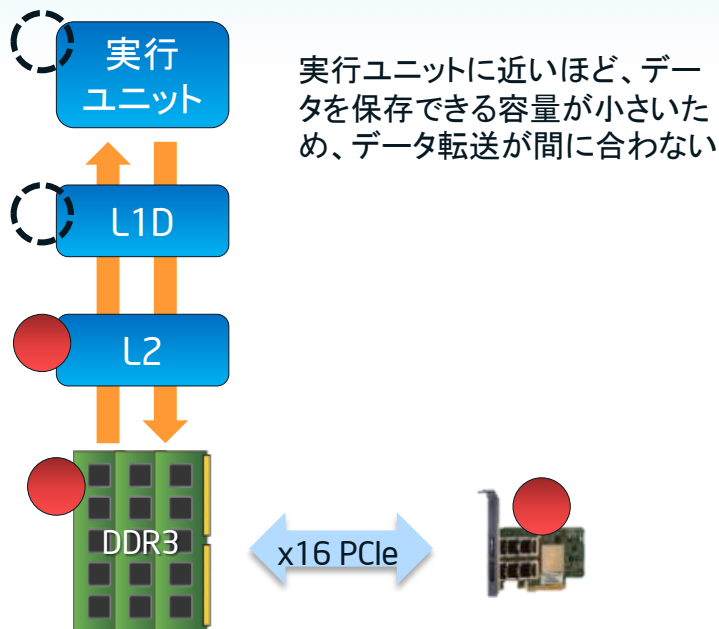
- ・データ・アライメント
- ・キャッシュ・ブロッキング
- ・連想性 (set associativity) による L1 の 4KB のアクセス間隔またはデータサイズ、L2 の 64KB のアクセス間隔などは遅延を招くため、パディングバイトを追加する

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

さまざまなボトルネックの原因を特定

キャッシュに関するボトルネック



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。

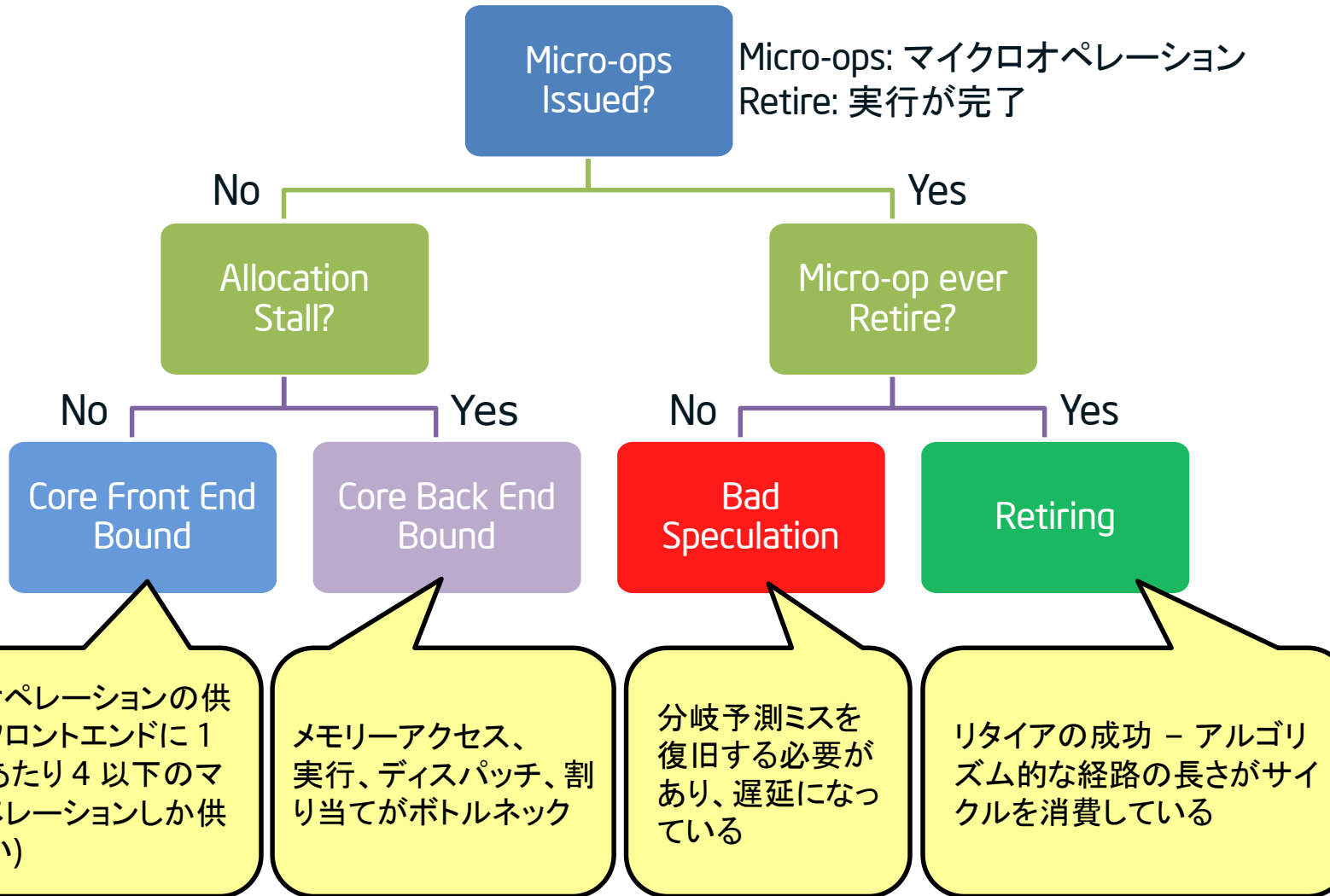
多発する分岐予測ミスがある箇所の分岐方法や処理方法を変更して、パフォーマンスを改善させる。

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向けには、インテル製マイクロプロセッサ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固有の最適化は、インテル製マイクロプロセッサ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 ロゴ、Intel Inside ロゴ、Intel Atom、Intel Atom Inside、Intel Core、Core Inside、Intel NetBurst、Intel Xeon Phi、Pentium、Xeon、Xeon Inside、VTune は、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。

Code the Future