インテル® ソフトウェア開発ツールで始める

並列アプリケーションのスケーラビリティーの問題を特定

応用編

2019 年 4 月
iSUS 編集長
すがわら　きよふみ

本日の予定

10:30 – 17:00

- コードの現代化と最適化 10:30 – 11:00
- 並列アプリケーションのスケーラビリティーの問題を特定 11:00 – 12:00

昼食

- インテル® コンパイラーによる高度な最適化 13:00 – 14:00
- インテル® VTune™ Amplifier 応用編 14:00 – 15:00

休憩

- インテル® Advisor 応用編 15:15 – 16:15
- インテル® Inspector 応用編 16:15 – 16:45
- まとめと Q/A 16:45 – 17:00

© 2019 iSUS *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。

並列アプリケーションの問題

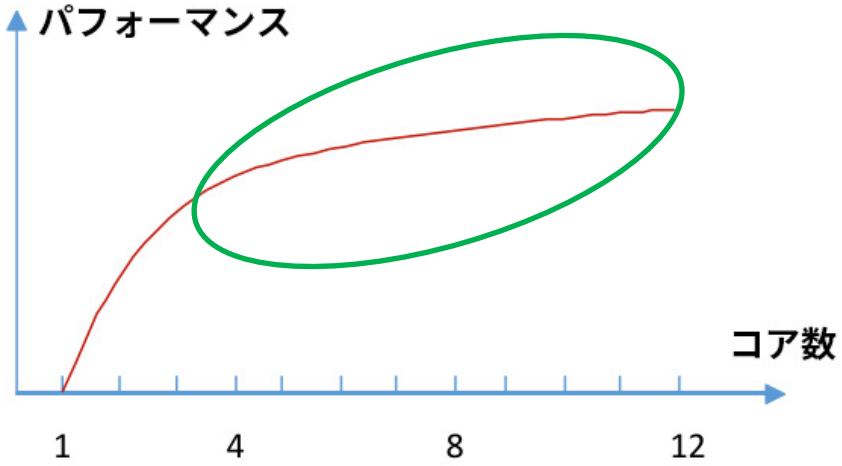
システムの計算コア数が増え続ける中、効率良く並列化されたソフトウェアでは、コア数の増加に伴ってパフォーマンスが直線的に向上することが望まれます。いくつかの要因により、**マルチコアシステムの並列性とスケーラビリティーが妨げられる**ことが分かっています

- **ロードインバランス**により、スレッドと CPU コアがアイドル状態になります
- **過度の同期**により、スピン待機やその他の非生産的なワークに CPU 時間が費やされます
- **並列ランタイム・ライブラリーのオーバーヘッド**が、適切でないライブラリー API の使用により発生します

これらの要因を排除することで、並列処理の効率が大幅に向上し、すべての CPU コアが有用なワークを実行するためビジー状態になります

© 2019 iSUS *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。

スケーラビリティーの問題



- STREAM や LINPACK のような最適なチューニングが行われているベンチマークでは、ほぼ直線的なスピードアップを達成できます

システムのコア数の増加に伴い (または、コア数の多い新しいシステムでコードを実行した場合に)、アプリケーションのパフォーマンスが直線的に向上しなかったり、並列パフォーマンスが伸び悩むことがあります

© 2019 iSUS *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。

トップダウンのパフォーマンス解析アプローチ

- **ほかのプロセスによってシステムが常にビジー状態でない:**
  リソースを消費するほかのアプリケーションやサービスに計算時間が費やされていることがあります

- **アプリケーションがシステムの I/O に依存していない:**
  例えば、ディスク、ほかのファイルシステム、ネットワーク・システムの処理が完了するのを待っていないか確認します

- **システムに十分な物理メモリー容量がある:**
  ハードディスク・ドライブとのスワップが頻発するのを防ぎます

**問題の修正は、ソフトウェアを最適化するよりも
ハードウェアで行ったほうが容易**

© 2019 iSUS *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。

理論値は達成可能なゴールなのか？

目標設定に有効なツール

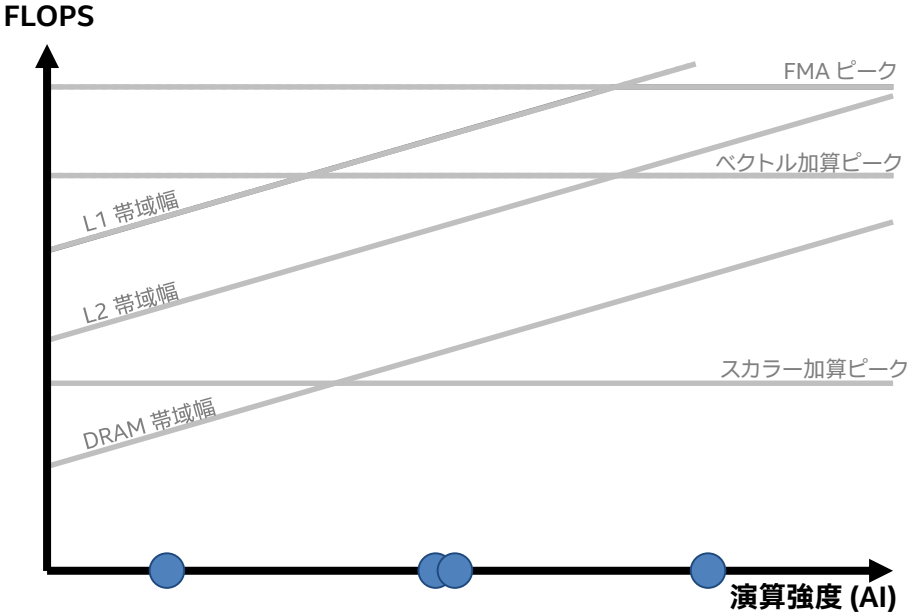
© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

キャッシュを考慮したルーフライン

インテル® Advisor

- データ収集前にインテル® Advisor は簡単なベンチマークを実行してハードウェアの能力を評価します
  - 計算の制限
  - メモリー帯域幅の制限
- これらがパフォーマンスの "ルーフ" となる
- ループと関数にはアルゴリズムがあり、特定の AI があります
- ループと関数のパフォーマンス (FLOPS) を測定します
- 最適化によりパフォーマンスは変化します。目標は最大限のパフォーマンスを達成することです

ビデオ: インテル® Advisor 2018/2019 のルーフライン解析



ルーフラインは最初にカリフォルニア大学バークレー校によって提案された:
Roofline: An Insightful Visual Performance Model for Multicore Architectures (英語), 2009
キャッシュを考慮したルーフラインはリスボン工科大学によって提案された:
Cache-Aware Roofline Model: Upgrading the Loft (英語), 2013

© 2019 iSUS *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。

アプリケーション・パフォーマンス・スナップショットに MPI を追加

MPI + OpenMP* + メモリーのデータを 1 個所にまとめる — インテル® VTune™ Amplifier

パフォーマンスの概要を簡単に素早く確認

- パフォーマンス・チューニングが必要か?

MPI と非 MPI アプリケーション †

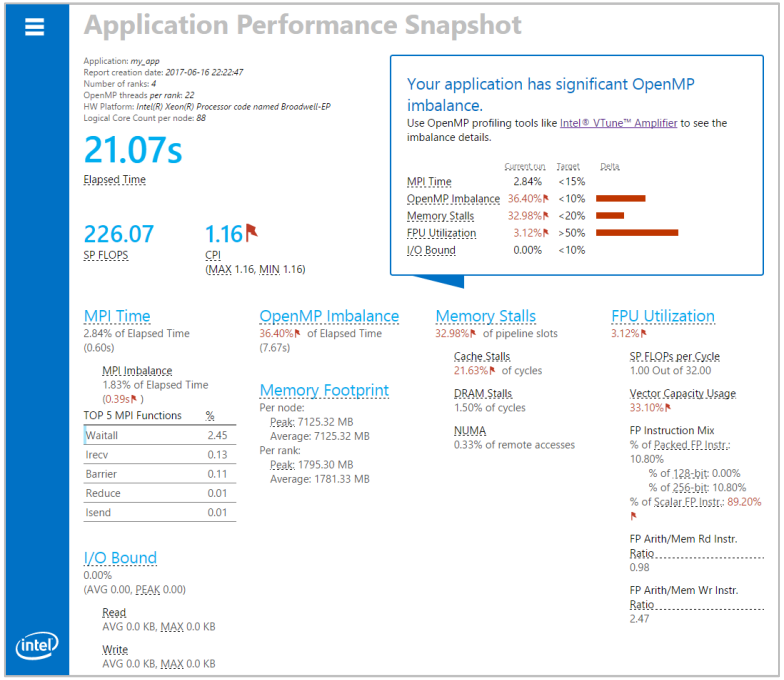
- シングル/マルチスレッドの分散 MPI
- 共有メモリー・アプリケーション

一般的な MPI 実装をサポート

- インテル® MPI ライブラリー
- MPICH と Cray* MPI

計算効率に関する豊富なメトリック

- CPU (プロセッサー・ストール、メモリーアクセス)
- FPU (ベクトル化メトリック)



† MPI は Linux* でのみサポート

© 2019 iSUS *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。

ワークロードとシステム構成のチューニング

インテル® VTune™ Amplifier

検出

- 構成の問題点
- 適切にチューニングされていないソフトウェア

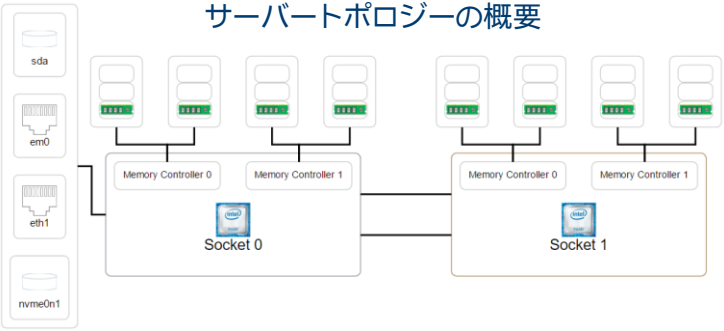
ターゲットユーザー

- インフラストラクチャー・アーキテクト
- ソフトウェア・アーキテクト/QA

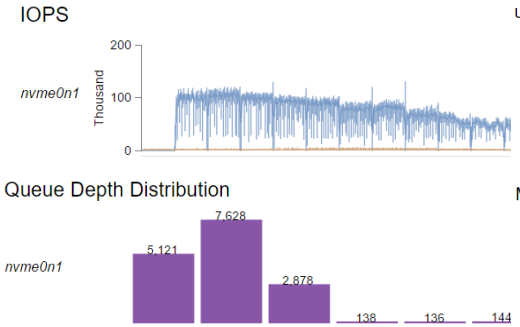
パフォーマンス・メトリック

- キャプチャー時間の拡張 (数分～数時間)
- 低オーバーヘッド – 粗粒度のメトリック
- OS/ハードウェア・パフォーマンス・カウンターのサンプリング
- スクリプトによる解析が簡単な RESTful API

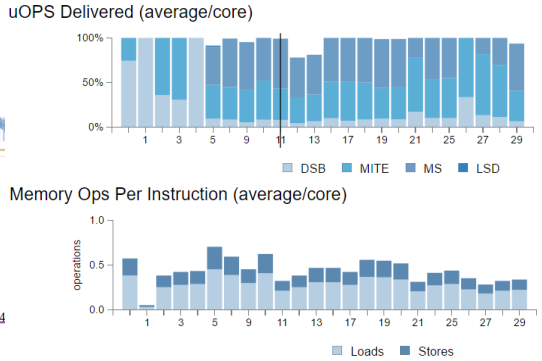
サーバートポロジーの概要



タイムラインとヒストグラム



コア間の比較



© 2019 iSUS *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。

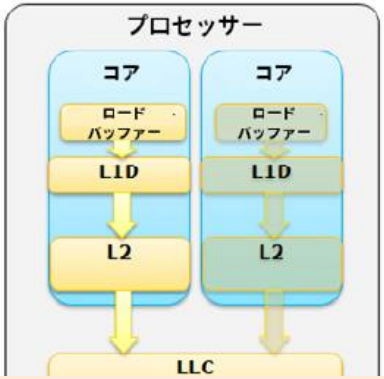
次のステップ

© 2019 iSUS *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。

次のステップ:メモリー・レイテンシーを確認

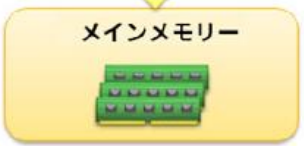
基本的に、データが遅延する理由は 2 つ

1. CPU の実行ユニットで実行される命令がデータを要求し、プリフェッチャーが適切に動作しない場合、メインメモリーやほかのキャッシュから CPU の L1D にータが読み込まれます。これにより、メモリー・レイテンシーの問題が生じます



2. プリフェッチャーが適切に動作し、データが事前に要求されても、転送インフラストラクチャーの処理能力が原因で CPU への転送が遅れることがあります。この場合、メモリー帯域幅の問題が生じます

複数のソースから複数の要求があった場合、両方の問題が発生する可能性があります



メモリー・レイテンシーの問題を解決するには、アドレスを使用してインクリメンタルにデータにアクセスします

メモリー帯域幅の問題を解決するには、データを再利用し、できる限りキャッシュに保持するようにします

© 2019 iSUS *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。

アプリケーションのスケーラビリティーを妨げる要因

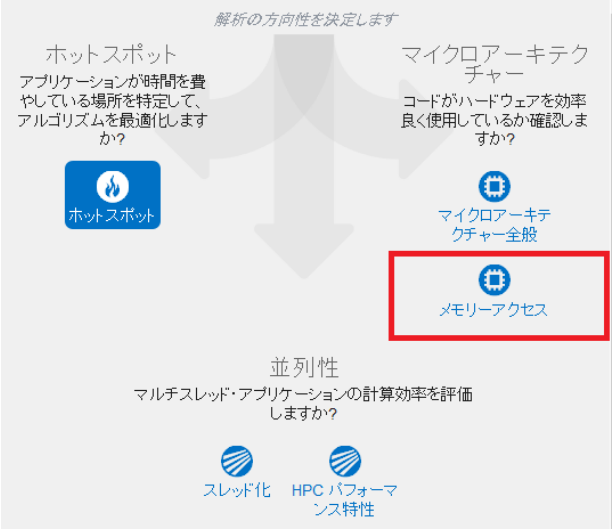
CPU でコードが効率良く実行されず、ほとんどのストールがメモリー依存であることが分かったら、問題の具体的な原因を明らかにする必要があります

メモリー・レイテンシー

?

メモリー帯域幅

インテル® VTune™ Amplifier のメモリーアクセス解析を使用して、メモリーの問題を詳しく調査



© 2019 iSUS *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。

例: 行列乗算アルゴリズムのナイーブ実装

```
void multiply2(int msize, int tidx, int numt,
          TYPE a[][NUM], TYPE b[][NUM], TYPE c[][NUM], TYPE t[][NUM])
{
     int i,j,k;
     // ループ交換
     for(i=tidx; i<msize; i=i+numt) {
        for(j=0; j<msize; j++) {
            for(k=0; k<msize; k++) {
                    c[i][j] = c[i][j] + a[i][k] * b[k][j];
}}}}
```

このシステムのパフォーマンス(理論値)

- メモリー帯域幅は 76.8GB/ 秒
- 1 秒あたりの倍精度 (DP) 浮動小数点演算の回数 (FLOPS)は 662GFLOPS/ 秒

スレッド数	実行時間(秒)	DP FLOPS (GFLOPS/秒)
4	208	7.7
8	102	15.1
16	59	26.8
32	42	37.8
72 HT	24	66.1

36 コア、インテル® Xeon® プロセッサー E5-2697 v4、2 ソケット @ 2300MHz

© 2019 iSUS *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。

インテル® VTune™ Amplifier の解析

S. Li.	Source	Cloc...	Inst... Ret...	CPI Rate	Front-... Bound	Bad Specu...	Back-... Bound	Retiring
66	void multiply1(int msize, int tidx, int numt, TYPE a[][NUM]							
67	{							
68	int i,j,k;							
69								
70	// Naive implementation							
71	for(i=tidx; i<msize; i=i+numt) {							
72	for(j=0; j<msize; j++) {	38,488...	35,...	1.077	1.1%	2.2%	76.6%	20.1%
73	for(k=0; k<msize; k++) {	71,300...	69,...	1.033	0.0%	0.0%	100.0%	14.7%
74	c[i][j] = c[i][j] + a[i][k] * b[k][j];	1,867,...	903...	2.067	0.7%	0.0%	87.1%	12.1%
75	}							
76	}							
77	}							
78	}							

リタイアした命令数が低く、CPI レートが高いことから、実際の制限からどれぐらいかけ離れているのか予測できます

© 2019 iSUS *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。

例: 行列乗算アルゴリズムのループ交換実装

```
void multiply2(int msize, int tidx, int numt,
          TYPE a[][NUM], TYPE b[][NUM], TYPE c[][NUM], TYPE t[][NUM])
{
     int i,j,k;
     // ループ交換
     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];
}}}}
```

絶対数はやや向上しましたが、まだ理想からはかけ離れています

スレッド数	実行時間(秒)	DP FLOPS (GFLOPS/秒)
4	208.8	7.8
8	103.3	15.1
16	58.8	26.4
32	38.4	40.5
72 HT	24.7	63.0

36 コア、インテル® Xeon® プロセッサー E5-2697 v4、2 ソケット @ 2300MHz

© 2019 iSUS *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。

インテル® VTune™ Amplifier の マイクロアーキテクチャー全般解析

CPU マイクロアーキテクチャー向けの別のトップダウン解析アプローチ

Function / Call Stack	Clockticks ▼	Instructions Retired	Front-End Bound	Bad Specul...	Back-End Bound	
					Memory Bound	Core Bound
multiply2	3,536,756,00...	1,160,060,2...	5.6%	0.1%	71.8%	14.3%

バックエンド依存値が減少

87.1%
↓
71.8%

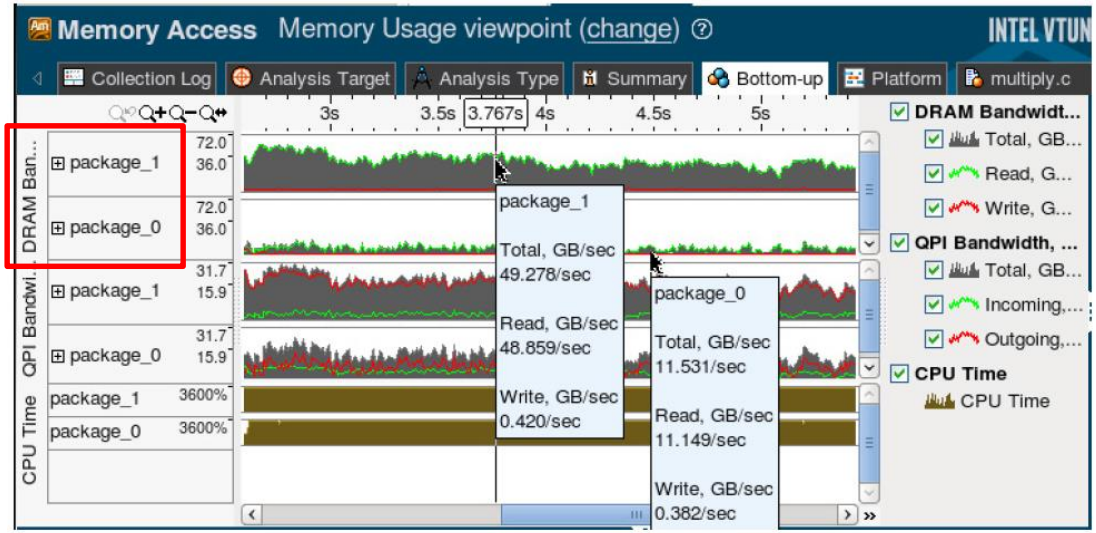
Back-End Bound							
Memory Bound							
L1 Bound	L2 Bound	L3 Bound	DRAM Bound				Store Bound
			Memory Bandwidth	Memory Latency			
				Local DRAM	Remote DRAM	Remote Cache	
7.9%	0.0%	33.3%	82.2%	29.0%	41.9%	0.0%	0.1%

- メモリー帯域幅メトリックが非常に高くなっているため、DRAM コントローラーと**インテル® QuickPath インターコネクト** がボトルネックになっていないか、メイン・データ・パスの帯域幅を確認
- データアクセスが連続したパターンにもかかわらず、L3 レイテンシーも大きくなっていることが分かります
- リモート DRAM レイテンシーが大きいことが分かります。これは、不均等メモリーアクセス (NUMA) による影響で、各ノードに対してデータの一部がリモート DRAM からフェッチされることを示しています

© 2019 iSUS *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。

NUMA の影響を調査 : 72 スレッドで実行

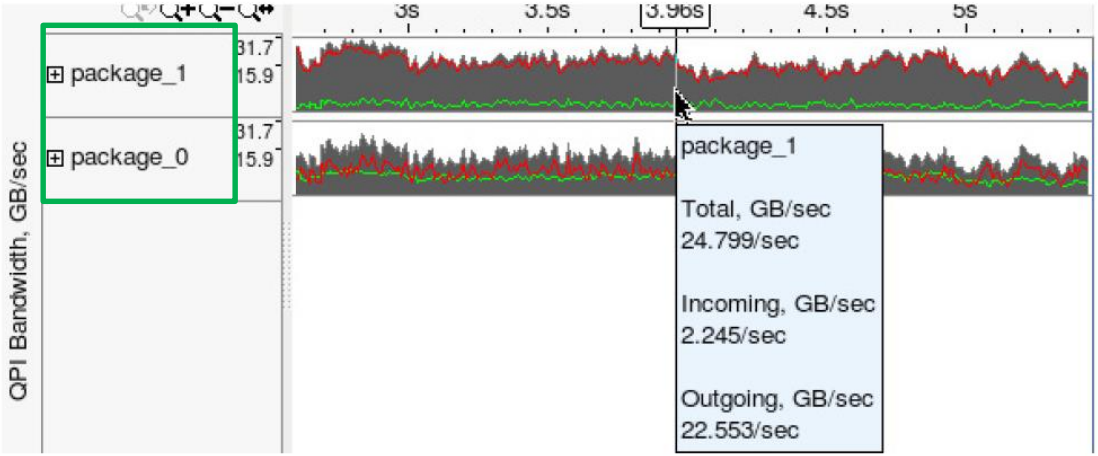
- package_1 の DRAM コントローラー のみでデータがロードされ、平均データ転送速度は約 50GB/ 秒、つまり最大帯域幅の約 2/3 であることが分かります
- package_0 のメモリー・コントローラー上にはほとんどトラフィックがありません



© 2019 iSUS *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。

NUMA の影響を調査 : 72 スレッドで実行

- package_1 の DRAM コントローラー のみでデータがロードされ、平均データ転送速度は約 50GB/ 秒、つまり最大帯域幅の約 2/3 であることが分かります
- package_0 のメモリー・コントローラー上にはほとんどトラフィックがありません
- インテル® QPI 出力レーンのデータ・トラフィックの半分は package_1 が占めています



プリフェッチャーによってリモート DRAM からフェッチされるデータと CPU コアによってリモート LLC からフェッチされるデータには、追加のレイテンシーがかかります

© 2019 iSUS *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。

CPU にスレッドをピニング

```
CreateThreadPool( … ){
   pthread_t ht[NTHREADS];
   pthread_attr_t attr;
   cpu_set_t cpus;
   pthread_attr_init(&attr);

   for(tidx=0; tidx<NTHREADS; tidx++) {
        CPU_ZERO(&cpus);
        CPU_SET(tidx, &cpus);
        pthread_attr_setaffinity_np(&attr, sizeof(cpu_set_t), &cpus);
        pthread_create(&ht[tidx],&attr,(void*)start_routine,(void*)&par[tidx]);
   }
   for (tidx=0; tidx<NTHREADS; tidx++)
        pthread_join(ht[tidx], (void **)&status);
}
```

- pthread_attr_setaffinity_np 関数を使用して、CPU アフィニティー・マスクを決定
- これにより、スレッドは CPU コアに固定されます

© 2019 iSUS *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。

データのブロック化：初期化関数

データ初期化関数では、配列は乗算関数で乗算される順序でスレッド間に分配されるべきです。初期化関数では、データ配列は行列のサイズをスレッド数で割った、サイズ **msize/numt** のチャンクに分割されます

```
InitMatrixArrays (int msize, int tidx, int numt, … ){
   int i,j,k,ibeg,ibound,istep;
   istep = msize / numt;
   ibeg = tidx * istep;
   ibound = ibeg + istep;
   for(i=ibeg; i<ibound; i++) {
        for (j=0; j<msize;j++) {
             a[i][j] = 1.0*i+2.0*j+3.0;
             b[i][j] = 2.0*i+1.0*j+3.0;
             c[i][j] = 0.0;
}}}
```


© 2019 iSUS *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。

データのブロック化 : matrix 関数

```
multiply2(int msize, int tidx, int numt, … ){
   int i,j,k,ibeg,ibound,istep;
   istep = msize / numt;
   ibeg = tidx * istep;
   ibound = ibeg + istep;
   for(i=ibeg; i<ibound; i++) {
        for(k=0; k<msize; k++) {
             for(j=0; j<msize; j++) {
                  c[i][j] = c[i][j] + a[i][k] * b[k][j];
}}}}
```

乗算関数でも同様の処理を行います

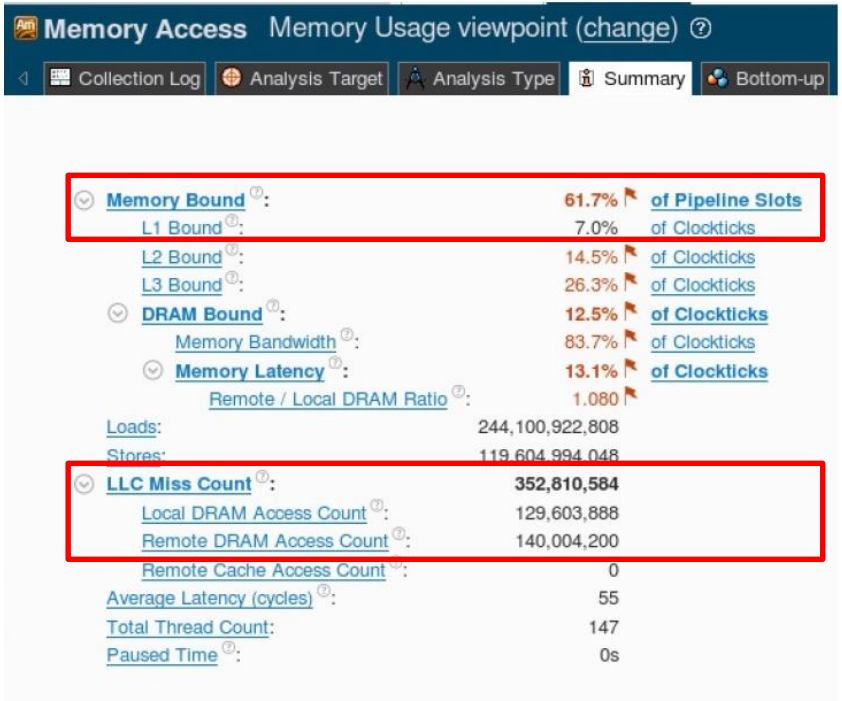
```
Threads #: 72 Pthreads
Matrix size: 9216
Using multiply kernel: multiply2
Freq = 2.30100 GHz
Execution time = 20.162 seconds
MFLOPS: 72826.877 mflops
```

ベンチマークの実行時間は、NUMA 非対応のバージョンとあまり変わりがありません

© 2019 iSUS *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。

インテル® VTune™ Amplifier のメモリーアクセス解析

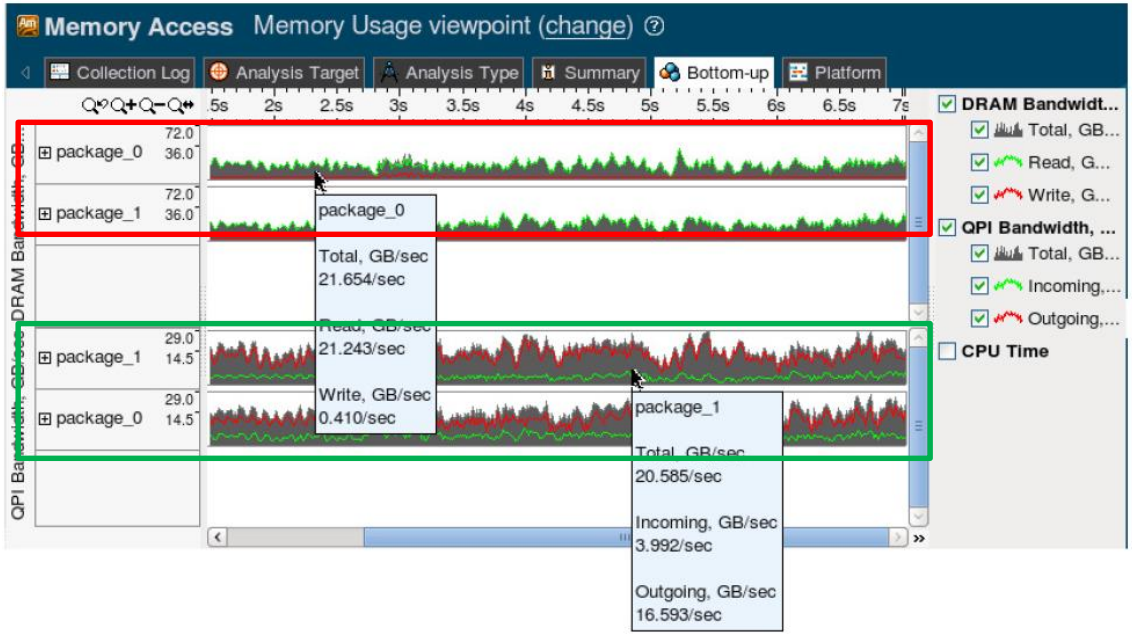
- [Summary] ページは、アプリケーションがまだメモリー依存であることを示しています (メモリーからのデータ・レイテンシーとデータ・トラフィックによるストールがあります)
- レイテンシーのほとんどは LLCによるもので、DRAM によるものはわずかです
- ローカルアクセスとリモートアクセスの比率が非常に高いことも分かります。これは、NUMA 対応のアプローチが適していないことを意味します



© 2019 iSUS *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。

DRAM コントローラーとインテル® QPI のトラフィックのタイムラインを確認

- DRAM からのデータストリームがピーク帯域幅の 30% に達することはほとんどありません
- インテル® QPI の帯域幅は各方向で約 90% に達しており、飽和状態といえます
- このシステムにおけるインテル® QPI の実際的な上限は 29.2GB/ 秒です



リモートアクセス (DRAM や LLC) は、メモリーブロック読み取りレイテンシーと CPU ストールを増加させます

© 2019 iSUS *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。

非効率的なリモートアクセスのデータ (行列) を特定

Top Memory Objects by Latency

This section lists memory objects that introduced the highest latency to the overall application execution.

Memory Object	Total Latency	Loads	Stores	LLC Miss Count
_mm_malloc (648 MB)	63.5%	112,553,776,512	25,600,384	339,210,176
_mm_malloc (648 MB)	36.2%	129,952,698,464	118,683,380,224	8,000,240
[Unknown]	0.2%	1,149,634,488	606,409,096	5,600,168
[Stack]	0.0%	120,803,624	201,603,024	0
_mm_malloc (648 MB)	0.0%	276,808,304	88,001,320	0
[Others]	0.0%	47,201,416	0	0

Grouping: Memory Object / Function / Allocation Stack

Memory Object / Function / Allocation Stack	Loads	Stores	LLC Miss Count	
			Local DRAM Acces...	Remote DRAM Access Count
[Unknown]	1,149,634,488	606,409,096	5,200,096	5,200,096
_mm_malloc (648 MB)	129,952,698,464	118,683,380,224	8,800,264	0
_mm_malloc (648 MB)	112,553,776,512	25,600,384	117,603,528	136,804,104
_mm_malloc (648 MB)	276,808,304	88,001,320	0	0

- 上位 3 つのメモリー･オブジェクト (割り当て関数によって表現) のうち、1 つのレイテンシーが最も大きく、多くのロード操作を行っていることが明らかです
- [Remote DRAM Access] から、LLC のリモート DRAMからデータを読み取っているのは、1 つのオブジェクトのみであることが分かります

© 2019 iSUS *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。

メモリー・オブジェクトのスタックを確認

ユーザーコードのスタックを確認して、インテル® VTune™ Amplifier の [Source] ビューでデータ割り当てのソース行にドリルダウン

Accesses (Memory Allocation)

Viewing 1 of 1 selected stack(s)

100.0% (41607124905 of 41607124905)

matrix.icc!_mm_malloc - [unknown source file]
matrix.icc!main+0x55 - matrix.c:106
matrix.icc!_start+0x28 - [unknown source file]

Source | Assembly | Assembly grouping:

S. Li.	Source
98	#else
99	buf1 = _aligned_malloc((sizeof (double))*NUM*NUM, 64);
100	buf2 = _aligned_malloc((sizeof (double))*NUM*NUM, 64);
101	buf3 = _aligned_malloc((sizeof (double))*NUM*NUM, 64);
102	buf4 = _aligned_malloc((sizeof (double))*NUM*NUM, 64);
103	#endif //ICC
104	#else // WIN32
105	buf1 = _mm_malloc((sizeof (double))*NUM*NUM, 64);
106	buf2 = _mm_malloc((sizeof (double))*NUM*NUM, 64);
107	buf3 = _mm_malloc((sizeof (double))*NUM*NUM, 64);
108	buf4 = _mm_malloc((sizeof (double))*NUM*NUM, 64);
109	#endif //WIN32
110	addr1 = buf1;
111	addr2 = buf2;
112	addr3 = buf3;
113	addr4 = buf4;

レイテンシーとロード操作の増加の原因は行列 b のデータにあります

© 2019 iSUS *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。

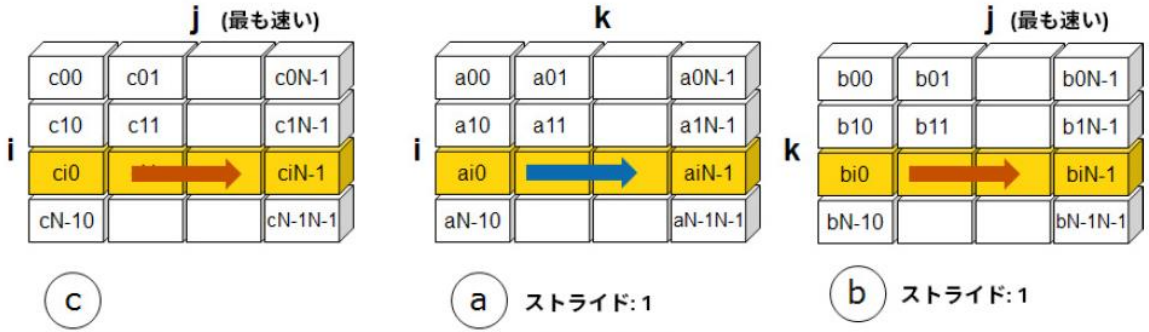
アルゴリズムの問題

転置行列のアルゴリズムを調査したところ、データ・アクセス・パターンの根本的な問題が見つかりました。行列 **a** の各行において、行列 **b** 全体をメモリーから読み取らなければなりません

```
for(i=0; i<N; i++){
        for(k=0; k<N; k++){
                for(j=0; j<N; j++){
  c[i][j]=c[i][j]+a[i][k]*b[k][j]; }}}
```

k++ ⇨ 2N 要素
i++ ⇨ $2N+N^2$ 要素

Float = 8 Byte, N = 9K
⇨ 72K / k++
⇨ 648MB / i++



© 2019 iSUS *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。

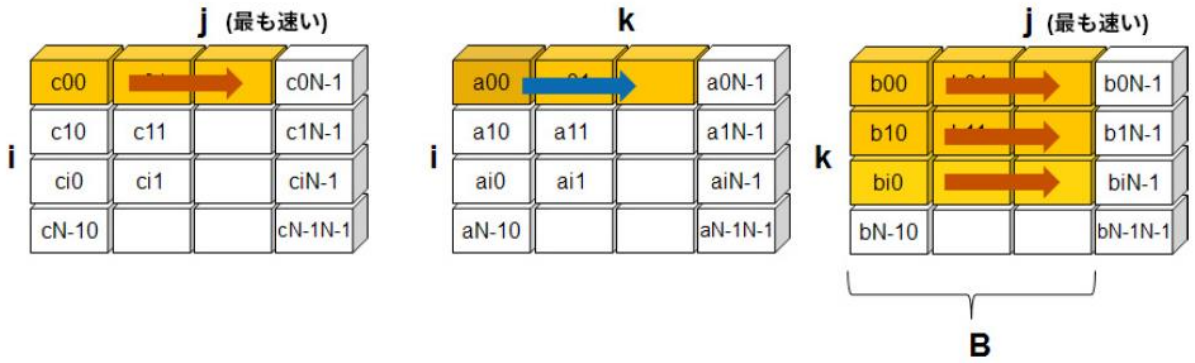
アルゴリズムの問題：データをブロック化

乗算アルゴリズムをさらに変更して、CPU ストールを排除することで、データ・レイテンシーを減らすことができます。ローカルソケットで実行中のスレッドが 3 つの行列のすべてのデータにアクセスできるようにします

```
for (i = ii; i < ii + B; i++) {
    for (k = kk; k < kk + B; k++) {
        for (j = jj; j < jj + B; j++) {
            c[i][j]  = c[i][j] + a[i][k] * b[k][j];
```

Float = 8 Byte
B = 64 要素
- L1D キャッシュに収まる

i++ ⇨ 2N + N^2 要素
N = 4K ⇨ 256K / i++
- L2 キャッシュに収まる



© 2019 iSUS *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。

結果を確認

キャッシュブロック変更後の結果から、NUMA による影響を修正しなくても、メモリー・レイテンシーが大幅に減り、高速に実行されるようになったことが分かります

```
./matrix.icc
Threads #: 72 Pthreads
Matrix size: 9216
Using multiply kernel: multiply3
Freq = 2.3 GHz
Execution time = 12.08 seconds
MFLOPS: 128710.367 mflops
```


© 2019 iSUS *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。

マイクロアーキテクチャー全般解析

- リタイアしたパイプライン・スロットが増加し、
- 残りの CPU ストールがメモリー依存とコア依存の実行で共有されることを示しています

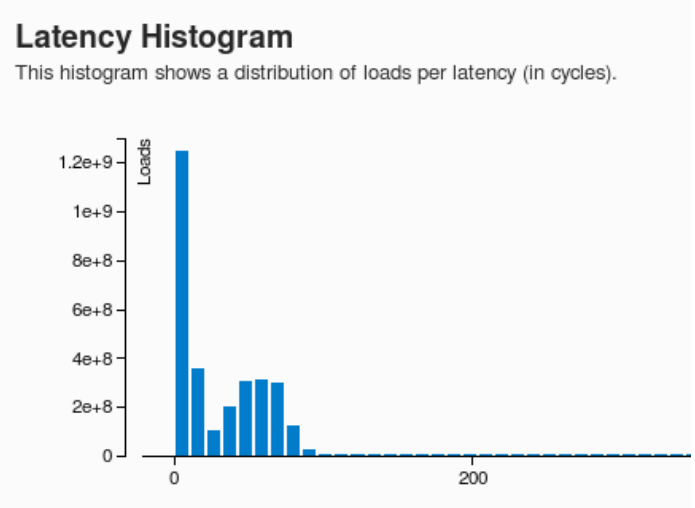
Clockticks:	2,268,184,100,000	
Instructions Retired:	1,851,470,100,000	
CPI Rate:	1.225	
MUX Reliability:	0.994	
Front-End Bound:	9.4%	of Pipeline Slots
Bad Speculation:	0.0%	of Pipeline Slots
Back-End Bound:	70.9%	of Pipeline Slots
Memory Bound:	45.6%	of Pipeline Slots
L1 Bound:	0.7%	of Clockticks
L2 Bound:	5.6%	of Clockticks
L3 Bound:	31.9%	of Clockticks
Contested Accesses:	0.0%	of Clockticks
Data Sharing:	0.0%	of Clockticks
L3 Latency:	100.0%	of Clockticks
SQ Full:	0.0%	of Clockticks
DRAM Bound:	3.0%	of Clockticks
Memory Bandwidth:	26.9%	of Clockticks
Memory Latency:	69.5%	of Clockticks
Local DRAM:	36.6%	of Clockticks
Remote DRAM:	6.5%	of Clockticks
Remote Cache:	0.0%	of Clockticks
Store Bound:	1.2%	of Clockticks
Core Bound:	25.3%	of Pipeline Slots
Retiring:	19.6%	of Pipeline Slots
Total Thread Count:	73	
Paused Time:	0s	

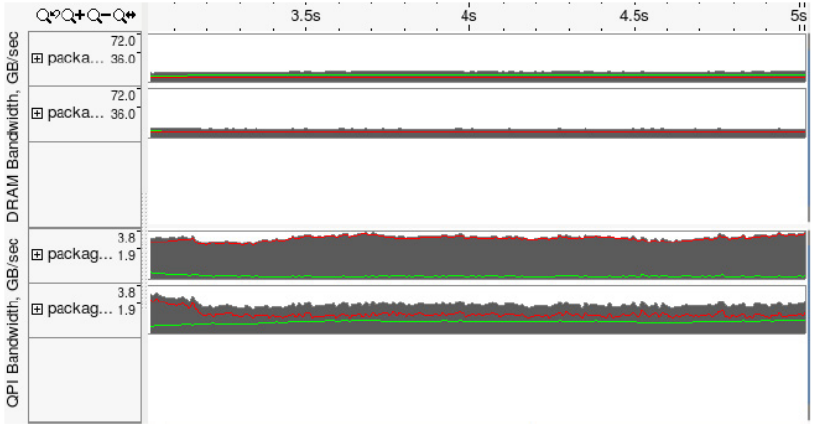
© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

分布図を確認

- レイテンシー分布図は、レイテンシーのほとんどが L2 アクセスに関連するもので、残りは 50 ~ 100 サイクルの LLC ヒット・レイテンシーであることを示しています

- 帯域幅タイムライン・ダイアグラムでは、データのほとんどがローカル DRAM から読み取られ、インテル® QPI トラフィックがわずかに増えたことが分かります





© 2019 iSUS *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。

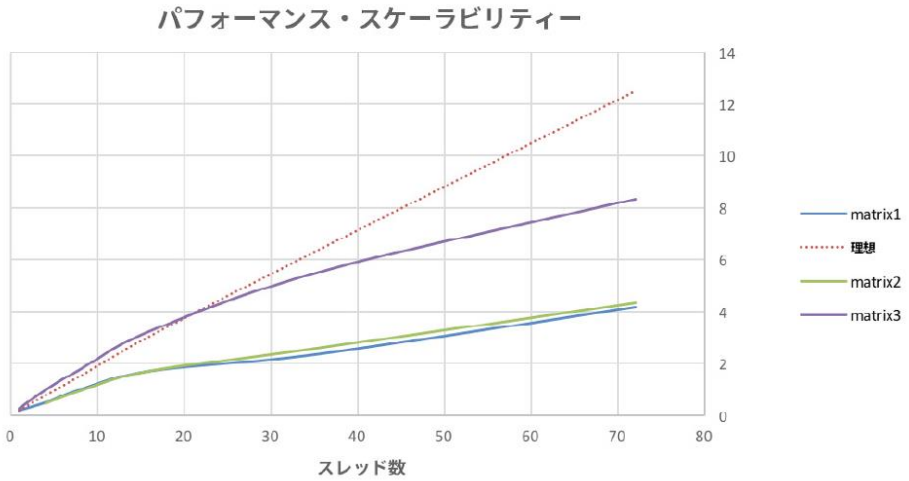
最終的なパフォーマンス結果とスケーラビリティー

最適化前

スレッド数	実行時間(秒)	DP FLOPS (GFLOPS/秒)
4	208	7.7
8	102	15.1
16	59	26.8
32	42	37.8
72 HT	24	66.1

最適化後

スレッド数	実行時間(秒)	DP FLOPS (GFLOPS/秒)
4	104.8	14
8	60.1	25
16	31.3	49
32	17.85	87
72 HT	12.8	128



スケーラビリティー・グラフに関する補足事項:

- matrix3 は、キャッシュ・ブロッキングにより理想的なスケーラビリティーを超えています。そのため、シングルスレッド・バージョンはナイーブ実装よりも高速に実行できます
- スレッド数が物理コア数と等しくなるまで、matrix3 は理想的なスケーラビリティーとほぼ同じです。ハイパースレッディングを導入しても、スケーリングは向上しません。

© 2019 iSUS *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。

まとめ

- 一部のメモリー・アクセス・パターンは、CPU マイクロアーキテクチャーの制限により、並列アプリケーションにおいてスケーラビリティーを妨げているように見えるかもしれません。これらの制限を回避するには、データの待機中に CPU ストールを引き起こすデータ配列を特定する必要があります
- インテル® VTune™ Amplifier のメモリー・アクセス・プロファイルを利用することで、最も大きな遅延を引き起こすデータ・オブジェクト、遅延の長さ (CPU クロック数)、データが配置されているキャッシュ・サブシステムのレベル、データ・オブジェクトの割り当てと遅延アクセスのソースコードを識別することができます
- この情報は、アルゴリズムを見直し、メモリー・アクセス・パターンを改善するのに役立つでしょう

© 2019 iSUS *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。

iSUS