インテルのキャッシュ・モニタリング・テクノロジー: 使用モデルとサンプルデータ (パート 3)

この記事は、インテル® デベロッパー・ゾーンに公開されている「Intel's Cache Monitoring Technology: Use Models and Data」の日本語参考訳です。

はじめに

インテルのキャッシュ・モニタリング・テクノロジー (CMT) 機能は、2014 年にリリースされたインテル® Xeon® プロセッサー E5-2600 v3 製品ファミリーで採用されました。CMT 機能は、(L3 キャッシュの占有状況から) 共有プラットフォーム・リソースの使用率を明らかにし、アプリケーション・プロファイリング、スケジュール、決定性、プラットフォームの可視性を向上して、ほかのアプリケーションのパフォーマンス低下を引き起こす共有リソースを過度に使用するアプリケーションを追跡します。 CMT はキャッシュの占有状況の詳細を明らかにすることで、リソース管理ソフトウェアがより良いサービスを提供できるようにします。

以前のブログの記事では、この機能のさまざまな側面について紹介しました。

- インテル® Xeon® プロセッサー E5-2600 v3 製品ファミリーの製品ページ: https://software.intel.com/en-us/articles/intel-xeon-e5-2600-v3-product-family
- パート 1: CMT の概要: http://www.isus.jp/article/mic-article/benefit-of-cache-monitoring/
- パート 2: RMID と CMT ソフトウェア・インターフェイスの説明: http://www.isus.jp/article/mic-article/software-visible-interfaces/

シリーズ第 3 弾となるこの記事では、ソフトウェア使用モデルについて述べ、いくつかの使用モデルで取得できるサンプルデータを提供します。

高レベルのモニタリング使用モデル

マルチプロセッサー・チップの共有リソースは、リソースを考慮したアプリケーション管理を有効にする、モニタリングと循環サイクルの割り当て制御フックにより管理できます (図 1)。リソース・モニタリングは可視性を高め、リソース使用率の追跡、利用可能なリソースに対するアプリケーションの稼働状況のプロファイリング、パフォーマンス/リソース反転の検出を可能にします。

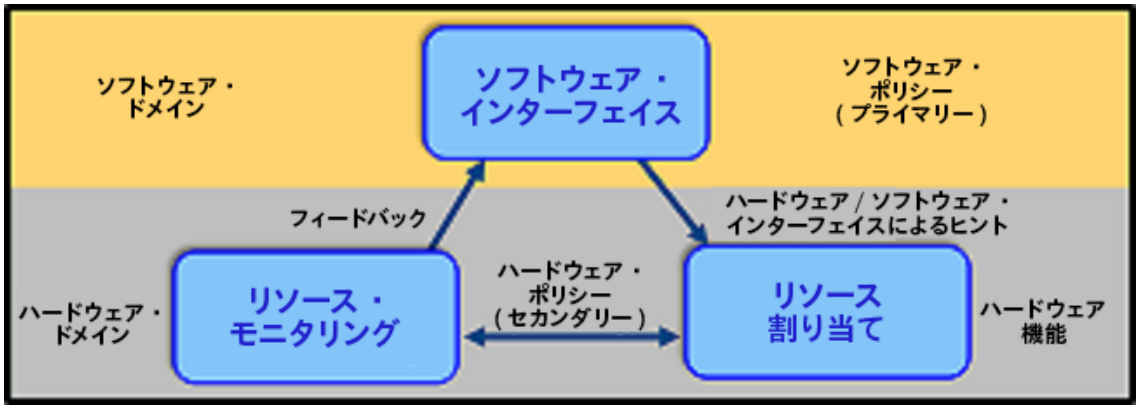

図 1. リソース・モニタリングは、データセンター環境においてより効率良いリソース割り当てを可能にする、共有リソース管理システムの重要なコンポーネントです。

インテルのキャッシュ・モニタリング・テクノロジー機能により、最終レベル (L3) キャッシュの占有状況をモニタリングできます。図 2 は、すべての使用モデルに共通のハードウェア/ソフトウェア・インターフェイス・モデルの概念図です。図の左側から開始し、最初のステップでは CPUID からプロセッサーで CMT 機能が有効かどうかを確認します。パート 2 [1] で述べたとおり、各サブレベルは、CMT サポートのレベルとパラメーターの詳細を提供します。これらは、サポートされるリソース・モニタリング ID (RMID) の数を含めプロセッサー世代により異なることがあります。

RMID は、CMT 機能を利用して、スレッド、アプリケーション、VM の追跡を可能にします。パート 2 [1] で述べたとおり、各スレッドに RMID を割り当てることができます。また、複数のスレッドに同じ RMID を割り当てることもできます。つまり、アプリケーションのすべてのスレッドまたは VM のすべてのアプリケーションを 1 つの RMID で追跡し、(ほかのアプリケーション/VM と並行に実行しながら動的にリアルタイムで) アプリケーション全体または VM 全体におけるキャッシュの利用状況を判断できます。

スレッドと RMID の関連付けは OS/VMM により処理され、スレッドへのスワップ時にスレッドごとのレジスター (IA32_PQR_ASSOC または略して PQR) に書き込まれます (図 2 のステップ 2)。ソフトウェアの対応状況とサポートについては、次の記事で詳しく述べます。

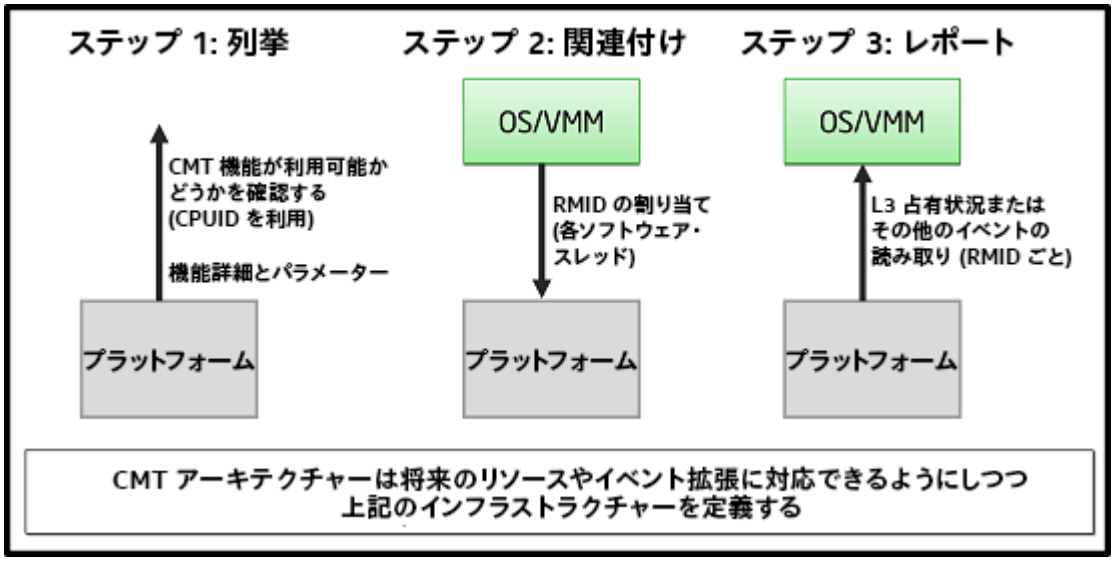

図 2. さまざまな CMT 使用モデルに共通する 3 ステップのプロセスは、列挙、関連付け、レポートです。

ソフトウェアによって定義された時間が経過すると、モデル固有レジスター (MSR) のペアを使用して RMID ごとのキャッシュの占有状況を読み取ることができます。

取得したモニタリング・データは、ソフトウェアによってスケーリングされ、後述するさまざまな目的に使用できます。

キャッシュ・モニタリング・テクノロジーの使用モデル

すべての使用モデルで、CMT は RMID を使用して多数のスレッド/アプリケーション/VM を動的に同時にモニタリングできます。

主な使用モデルは次のとおりです。

- リアルタイム・プロファイリング: アプリケーション・パフォーマンスとキャッシュの占有状況の把握 (図 3)

- キャッシュの容量不足のアプリケーションの検出 (キャッシュの容量不足を緩和することでパフォーマンスを向上)
- システム・スループットを向上するキャッシュを考慮した高度なスケジュール

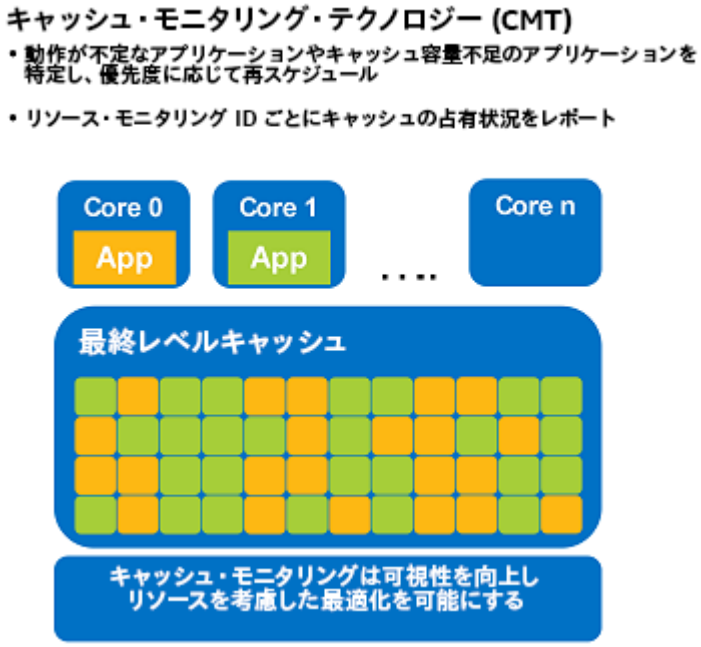

図 3. キャッシュ・モニタリング・テクノロジーによるプロファイリングの例

このほかにも、CMT には次のような使用モデルがあります。

- 異常検出とソフトウェアのチューニング/最適化のための長時間の動的アプリケーション・プロファイリング
- シミュレーターを使用しない高精度で正確なキャッシュの利用状況の測定
- キャッシュ競合の検出と測定 (同時に実行中のアプリケーション/VM からキャッシュ不足のアプリケーション/VM の検出を含む)
- SLA のパフォーマンスのモニタリング
- クラスターへ新しいアプリケーションを追加する最適な方法の検証
- 課金/費用管理
- 管理: データセンター内の効率を評価するため、データを集計しデータセンター管理者に提供

次に、CMT を利用したリアルタイム・プロファイリング使用モデルの詳細な例を示します。

キャッシュ・モニタリング・テクノロジーのサンプルデータ: アプリケーション・プロファイリング

CMT を使用することで、プラットフォームでアプリケーションを実行しながらモニタリングできます。図 4 に示す仮想化されていないケースでは、14 コアのインテル® Xeon® プロセッサー E5-2600 v3 ベースのシステムで複数のアプリケーションが実行され、RMID は各コアにピニングされています。アプリケーションを実行すると、各アプリケーションのキャッシュの占有状況が定期的にサンプリングされます。図 4 から、オペレーティング・システムのタスクの変動に伴って、占有状況 (緑色の線) が変わっているのが分かります。グラフの中央で、1 つのコアで新しいメモリー・ストリーミング・アプリケーションが呼び出され、L3 キャッシュをすべて占有してから終了しています。CMT を利用することで、このアプリケーションを検出し、ほかのより重要なアプリケーションの動作に影響する場合は、このアプリケーションを別のプロセッサーやノードに移動することが可能になります。このアプリケー

ションが単にリソースを多用するアプリケーションであり、優先度が高い場合は、CMT で一定期間にわたってアプリケーションの実際のキャッシュの利用状況を測定することができます (図 5。詳細は後述します)。

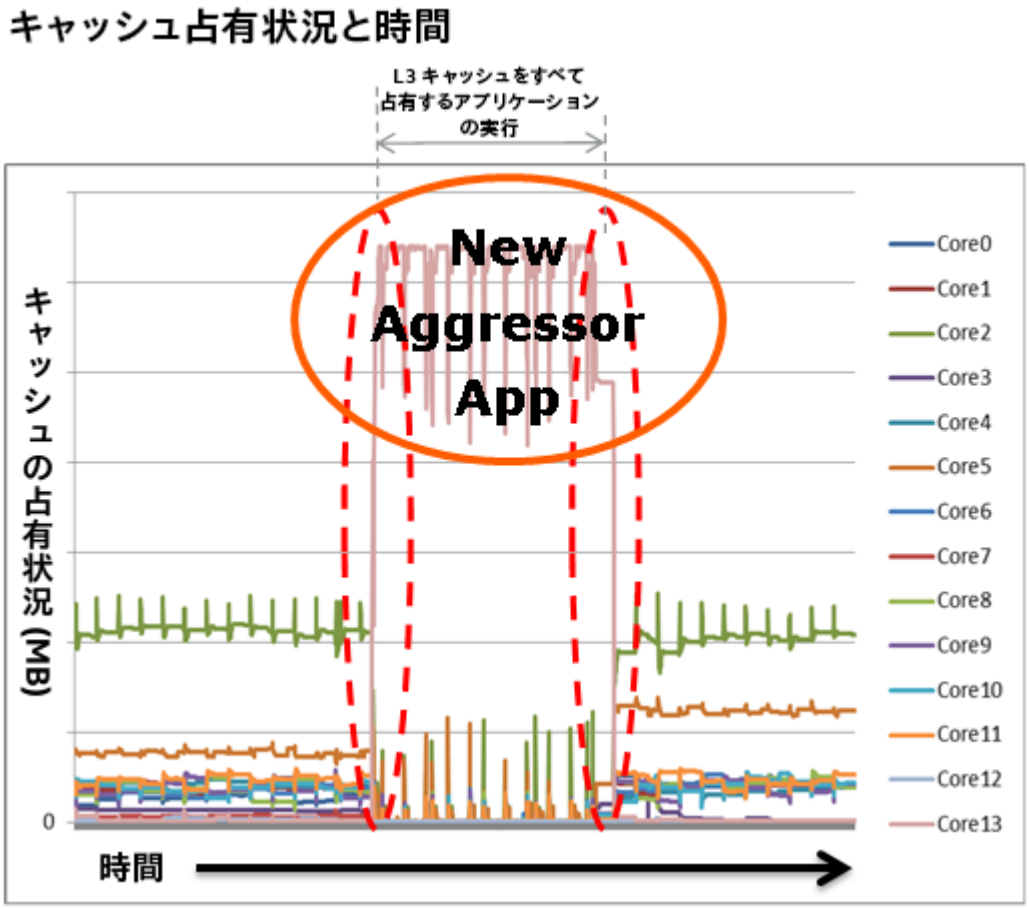

図 4. 各コアに 1 つの RMID をピニングした場合の CMT の結果を時系列で表したグラフです。メモリー・ストリーミング・アプリケーションによりグラフの中央でキャッシュの占有率が大幅に上昇しています。このアプリケーションは、すべてのキャッシュを占有してから終了します。このグラフは、インテル® Xeon® プロセッサー E5 v3 ベースのサーバーで収集したデータを基に生成されています。

キャッシュ・モニタリング・テクノロジーを利用することで、アプリケーションの稼働状況、特にキャッシュを多用するアプリケーションを動的に測定し、記録することができます (図 5)。

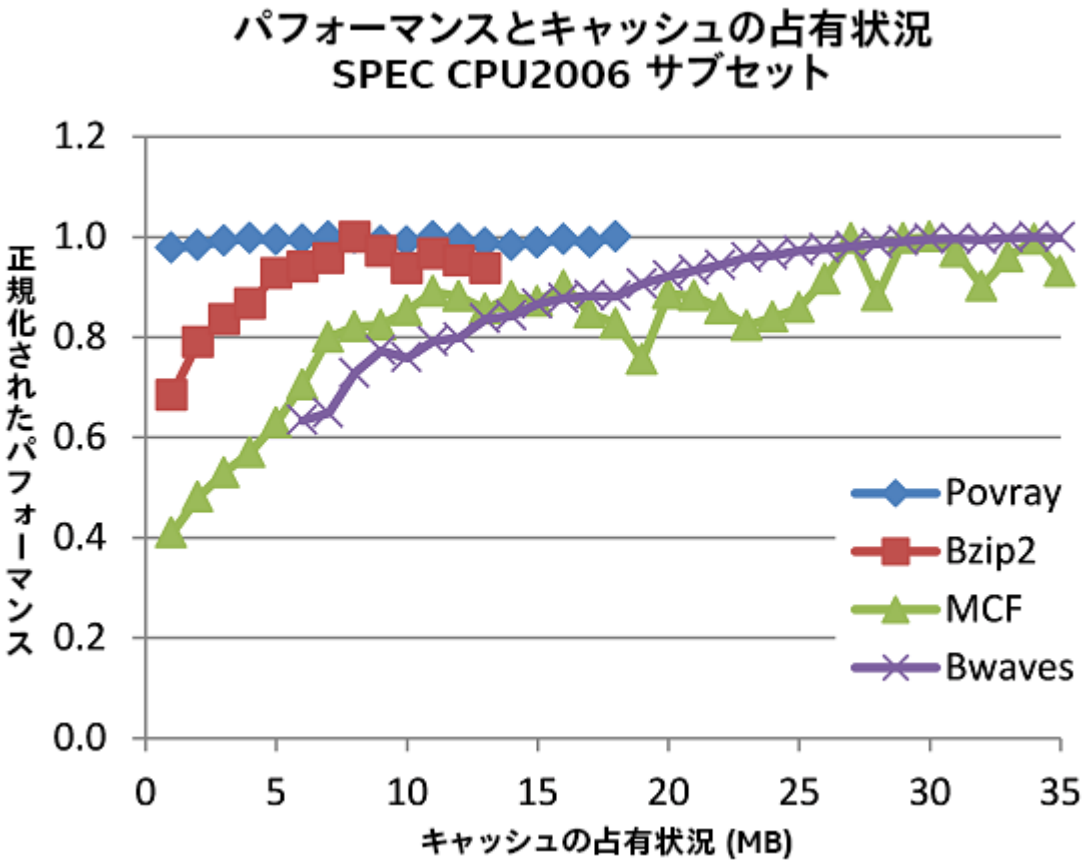

図 5. CMT を使用して生成した SPEC* CPU2006 のさまざまなアプリケーションのキャッシュの利用状況のグラフです。

図 5 は、実際のシステムで、各アプリケーションをさまざまなバックグラウンド・アプリケーションと並行に実行し、定期的に占有状況データをサンプリングして、各範囲のすべてのサンプルの平均を集計しています。ここでは、範囲にキャッシュの占有状況 0-1MB、1-2MB、2-3MB、... (単純な "バケツ" 手法) を使用しています。そして、正規化されたアプリケーション・パフォーマンスとキャッシュの占有状況を描画し、各アプリケーションのキャッシュの利用状況を明確に示しています。図のように、レイトレーシング・アプリケーション povray はキャッシュに影響されません (キャッシュサイズに関係なくパフォーマンスがほぼ一定です)。bzip2 アプリケーションは、L3 キャッシュが 8MB までの場合にキャッシュにやや影響されます (大体これぐらいのキャッシュを bzip2 に提供できるようにアプリケーションをスケジュールすべきです)。残り 2 つのアプリケーション (mcf と bwaves) はキャッシュに大きく影響されるため、パフォーマンスを最大限に引き出すには、できるだけ多くのキャッシュを利用できるようにすべきです。mcf の動作は変動性が高く、マルチフェーズのため、mcf データにはいくつかのノイズが含まれていますが、キャッシュの利用状況の評価には問題ありません。より正確な特性を把握する必要がある場合は、サンプリング期間を長くすると良いでしょう。

アプリケーションのキャッシュの利用状況が曲線適合し (通常は次の形式の対数: y = 係数 * ln(x) + 定数)、相関係数がある程度大きい場合、アプリケーション全体のキャッシュの利用状況は係数と定数のペアで表し、格納することができます。

より高度な解析は、曲線適合を適用後に行うことができます。例えば、占有状況に対するパフォーマンスと占有状況 (キャッシュの利用状況) 曲線の微分係数から、キャッシュ占有状況単位ごとのキャッシュの利用状況の曲線を求めることができます。しきい値 (図 6) を追加することで、指定されたアプリケーションのキャッシュの利用状況が正確に生成され、チューニングにより得られる最適なキャッシュの動作ポイントを特定できます。例えば、L3 キャッシュを 1MB 追加してもアプリケーション・パフォーマンスが 1% しか向上しないポイントのように、最適

なキャッシュの動作ポイントを定義できます。このようなしきい値は、動的スケジュールやキャッシュ不足のアプリケーションの検出に使用されます。

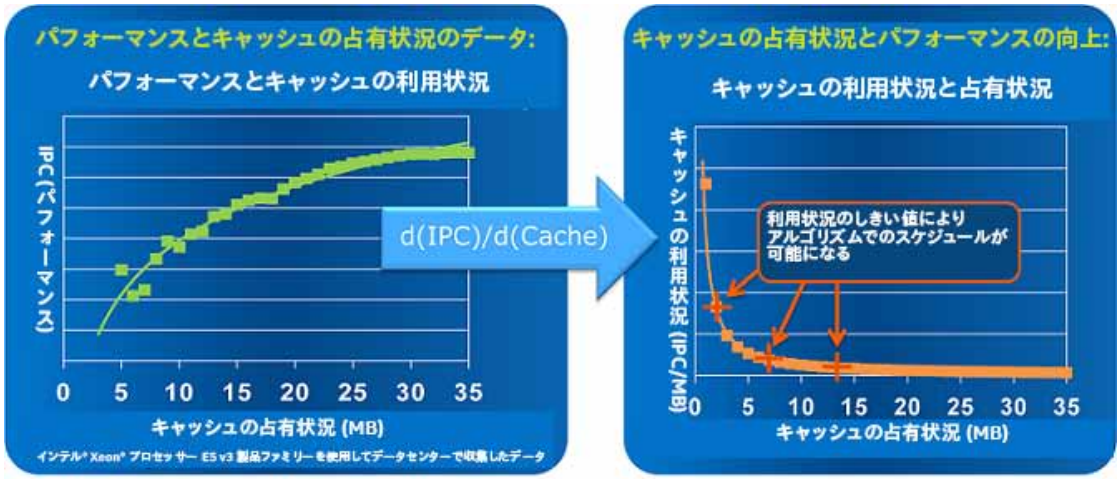

図 6. 曲線適合によるアプリケーションのキャッシュの利用状況(左)、および占有状況に対する微分係数から生成されるキャッシュの利用状況と占有状況 (右) のグラフです。単純なしきい値とともに使用することで、アルゴリズムの観点から最適なキャッシュの動作ポイントを判断できます。この例では、アプリケーション・パフォーマンスの代わりにサイクルあたりの命令数 (IPC) が使用されています。

アプリケーションの最適なキャッシュの動作ポイントを動的に判断するには、CMT と CMT により可能な新しい使用モデルが不可欠です。CMT を使用しないでこのデータを収集するには、シミュレーションを行うか (大きく変動するデータセンター環境では現実的ではありません)、推定法を使用します (一般に不正確で、データセンターにおいてワークロードの相互干渉が生じるため現実的ではありません)。

さまざまなアプリケーションで収集した占有状況曲線を利用して、アプリケーションの長期にわたる履歴を記録し、ソケット間のスケジュールを最適化できます。例えば図 7 の左のグラフのように、2 つの計算負荷の高いアプリケーションが 1 つプロセッサーに割り当てられており、ワーキングセットが小さい場合 (例えば、L3 キャッシュ 35MB のうち 4MB を使用する場合)、ソケット間でアプリケーションを再分配し、L3 キャッシュの使用率を最適化したり、パフォーマンスを向上できます (または、動的に移動する代わりに、次回アプリケーションを実行するときのために格納して、NUMA メモリーイメージの留意事項を単純化します)。

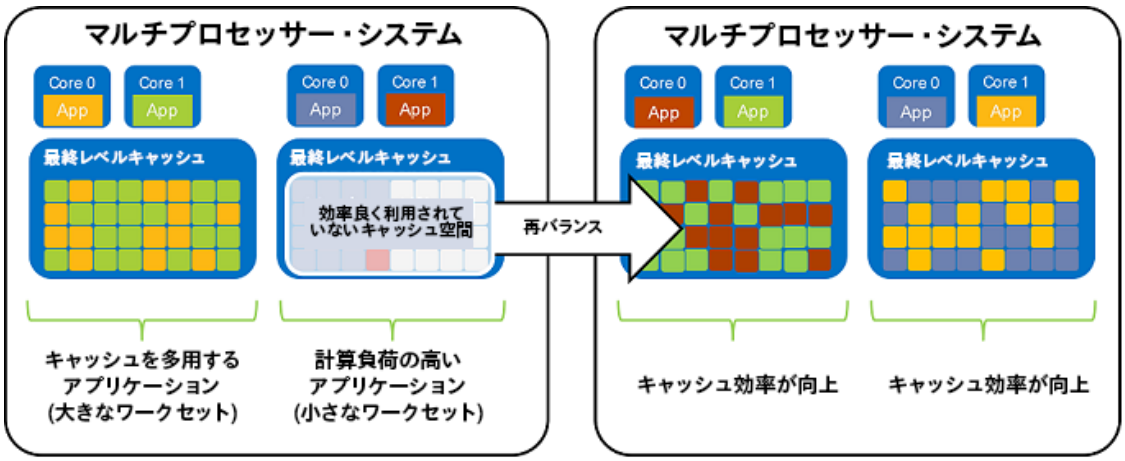

図 7. CMT を使用して、キャッシュ使用率を最適化するためプロセッサー間でアプリケーションを再配分します。

上記のサンプル・プロファイリング・データは、数ある使用モデルのうちの 1 つです。

結論

インテルのキャッシュ・モニタリング・テクノロジー (CMT) 機能は、スレッド、アプリケーション、VM、またはそれらの組み合わせを、さまざまなソフトウェア使用モデルに合わせて柔軟な方法で同時に追跡できます。収集したモニタリング・データは、アプリケーション・プロファイリング、キャッシュの利用状況の測定、キャッシュ競合の検出、SLA のパフォーマンスのモニタリング、キャッシュの容量不足のアプリケーションの検出、高度なキャッシュを考慮したスケジュール、新しいアプリケーションを追加する最適な方法の検証、課金/費用管理、その他のさまざまなリソースを考慮したスケジュールの最適化に使用できます。データセンター内の効率を評価し、ソフトウェアの最適化に役立てるため、データを集計しデータセンター管理者に提供することもできます。

以前の記事へのリンクは、冒頭にあります。次の記事では、ソフトウェアの対応状況、ツール、OS/VMM サポートについて取り上げます。

参考文献

[1] http://www.isus.jp/article/mic-article/software-visible-interfaces/

著者

Andrew Herdrich
インテル・ラボの研究員として、2008 年からキャッシュ・モニタリング・テクノロジーや将来のスレッド競合緩和テクノロジーの設計を担当し、最近では高度な将来のアーキテクチャーや NFV 向けの IA の最適化に取り組んでいます。インテル・ラボに配属される前は、Merom †、Nehalem †、Westmere † CPU や第 1 世代 Knights † 製品に携わっていました。

†開発コード

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください