NUMA NUMA NUMA !!

Part1 インテル® VTune™ Amplifier で NUMA の振る舞いを観察

2019 年 9 月

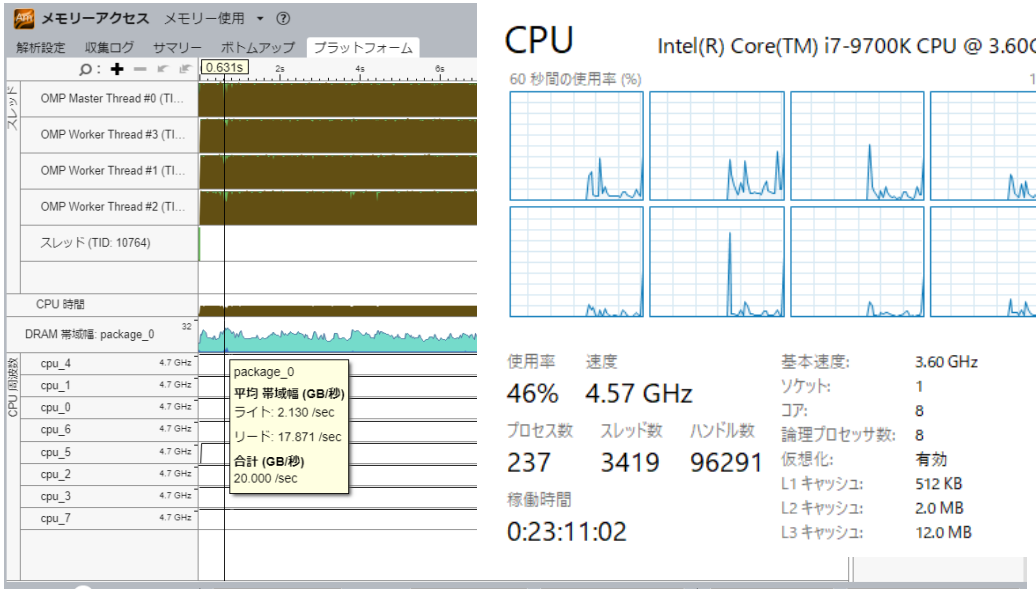
すがわら　きよふみ

内容

- シングルソケットのシステムで実行
- 同じバイナリーを NUMA システムで実行
- 振る舞いを確認

シングルソケット環境で実行

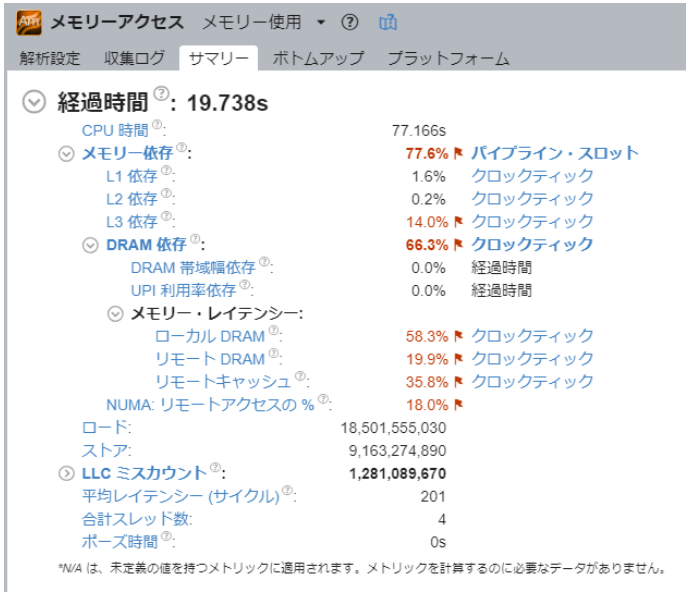
行列の乗算加算を行う並列アプリケーション

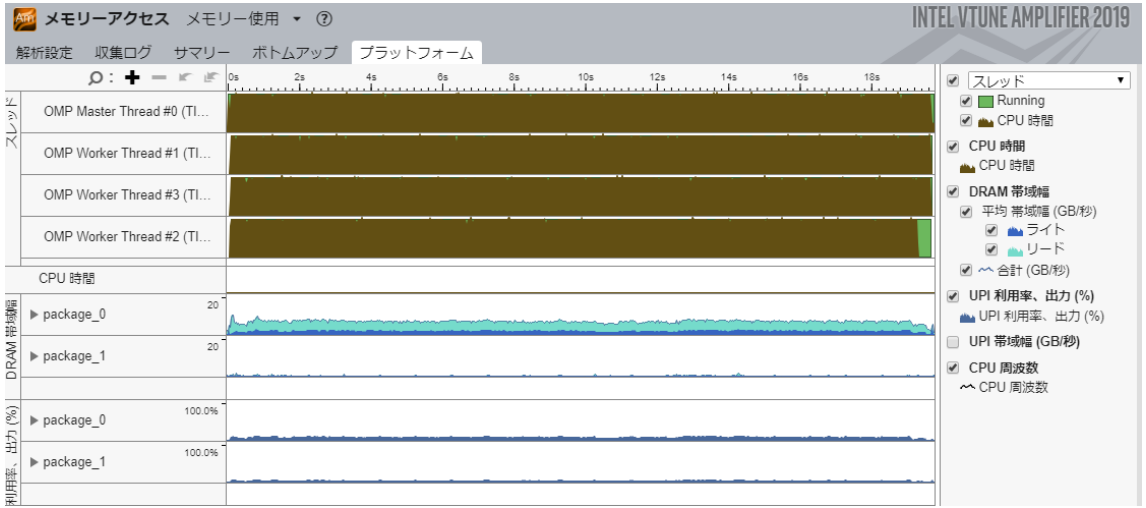


- キャッシュは有効に利用されておらず、DRAM アクセスが多いことが分かります

同じバイナリーを NUMA システムで実行

インテル® Xeon® Platinum 8276 (28 コア) プロセッサー X 2





- UPI 通信が発生し、package_0 の DRAM がアクセスされています
- 4 スレッドなのになぜ NUMA ノードをまたいでスレッドが生成されているか

ソースを確認

```
void multiply1(int msize, int tidx, int numt, TYPE a[][NUM], TYPE
b[][NUM], TYPE c[][NUM], TYPE t[][NUM])
{
    int i,j,k;

#pragma omp parallel for proc_bind(spread)
    for(i=0; i<msize; i++) {
        for(j=0; j<msize; j++) {
            for(k=0; k<msize; k++) {
                c[i][j] = c[i][j] + a[i][k] * b[k][j];
            }
        }
    }
}
```

ソース内でアフィニティーが設定されている

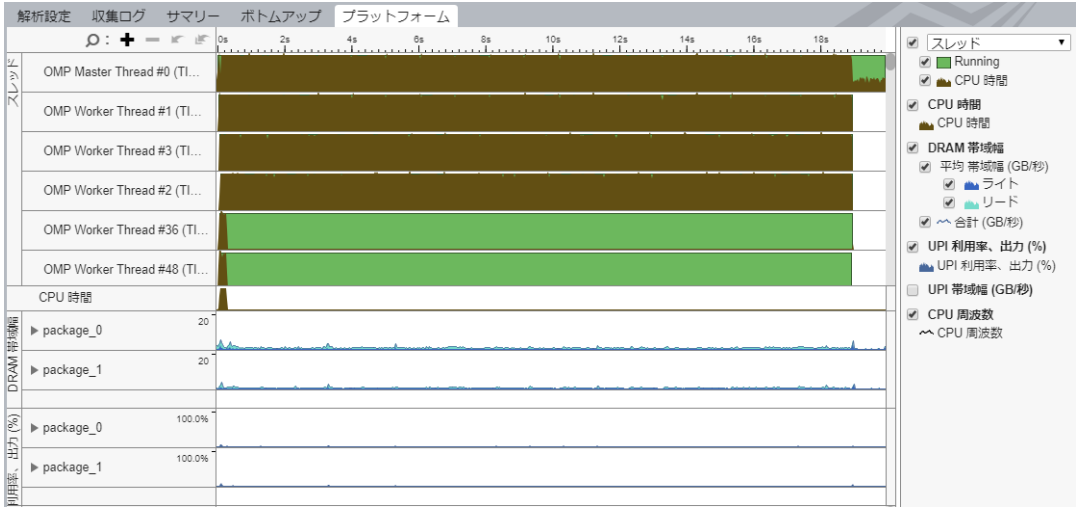
対処方法: その 1

解析設定　収集ログ　サマリー　ボトムアップ　プラットフォーム

経過時間: 19.916s

CPU 時間:	99.431s	
メモリー依存:	64.8%	パイプライン・スロット
L1 依存:	0.4%	クロックティック
L2 依存:	0.4%	クロックティック
L3 依存:	58.4%	クロックティック
DRAM 依存:	2.6%	クロックティック
DRAM 帯域幅依存:	0.0%	経過時間
UPI 利用率依存:	0.0%	経過時間
メモリー・レイテンシー:		
ローカル DRAM:	7.5%	クロックティック
リモート DRAM:	1.5%	クロックティック
リモートキャッシュ:	3.6%	クロックティック
NUMA: リモートアクセスの %:	11.8%	
ロード:	26,635,799,050	
ストア:	11,501,345,030	
LLC ミスカウント:	147,010,290	
平均レイテンシー (サイクル):	110	
合計スレッド数:	112	
ポーズ時間:	0s	

*N/A は、未定義の値を持つメトリックに適用されます。メトリックを計算するのに必要なデータがありません。



- リモートアクセスの問題は減少している
- Package_0 と Package_1 の DRAM が均等にアクセスされている
- UPI 通信はほとんど発生していない

どのように対処した？

```
void init_arr(TYPE row, TYPE col, TYPE off, TYPE a[][NUM])
{
  int i,j;

  #pragma omp parallel for proc_bind(spread)
  for (i=0; i< NUM;i++) {
    for (j=0; j<NUM;j++) {
      a[i][j] = row*i+col*j+off;
    }
  }
}
```

初期化ルーチンも OpenMP* で並列化した

プログラムの振る舞いを推測

基本速度:	2.20 GHz
ソケット:	2
コア:	56
論理プロセッサ数:	112
仮想化:	有効
L1 キャッシュ:	3.5 MB
L2 キャッシュ:	56.0 MB
L3 キャッシュ:	77.0 MB

- このプロセッサーの L3 キャッシュの容量は、33.5 MB です
- 一方のプロセッサーのみで初期化すると、書き込まれた配列は L3 キャッシュに収まりません
- 両方のプロセッサー上でスレッドが初期化を行うことで、77MB の L3 キャッシュを利用できます

- ただしこの方法は、初期化と計算のスレッド数が異なったり、異なるワークシェアのスケジュールを行うと効果が得られない可能性があります
- また、これは Windows* 固有の振る舞いで Linux* では、メモリー割り当ての方法が異なるため同じ動作になるとは限りません

まとめ

- NUMA システムでは、プログラムはシングル・ソケット・システムでは予想されなかった振る舞いをすることがあります
- マルチスレッド・アプリケーションが、どのようにメモリーを確保しアクセスすることを知るのは重要です
- これらの課題は、インテル® VTune™ Amplifier などの優れたツールを利用してプログラムの振る舞いを確認することで、その後のステップを見極めることができます

参考資料

- NUMA 向けのアプリケーションの最適化
  https://www.isus.jp/hpc/optimize-for-numa/
- NUMA ハードウェアによるパフォーマンスの向上
  https://www.isus.jp/hpc/performance-improvement-opportunities-with-numa-hardware/
- 等方性 3 次元有限差分 (3DFD) 波動方程式コード向けの NUMA を理解する
  https://www.isus.jp/products/vtune/understanding-numa-for-3d-finite-difference/
- マルチスレッド開発ガイド: 3.5 NUMA 向けのアプリケーションの最適化
  https://www.isus.jp/products/psxe/intelguide-3-5/


iSUS