インテル® System Studio 2015 のインテル® Energy Profiler を使用する

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「How to use the Intel® Energy Profiler in Intel® System Studio 2015」の日本語参考訳です。

はじめに

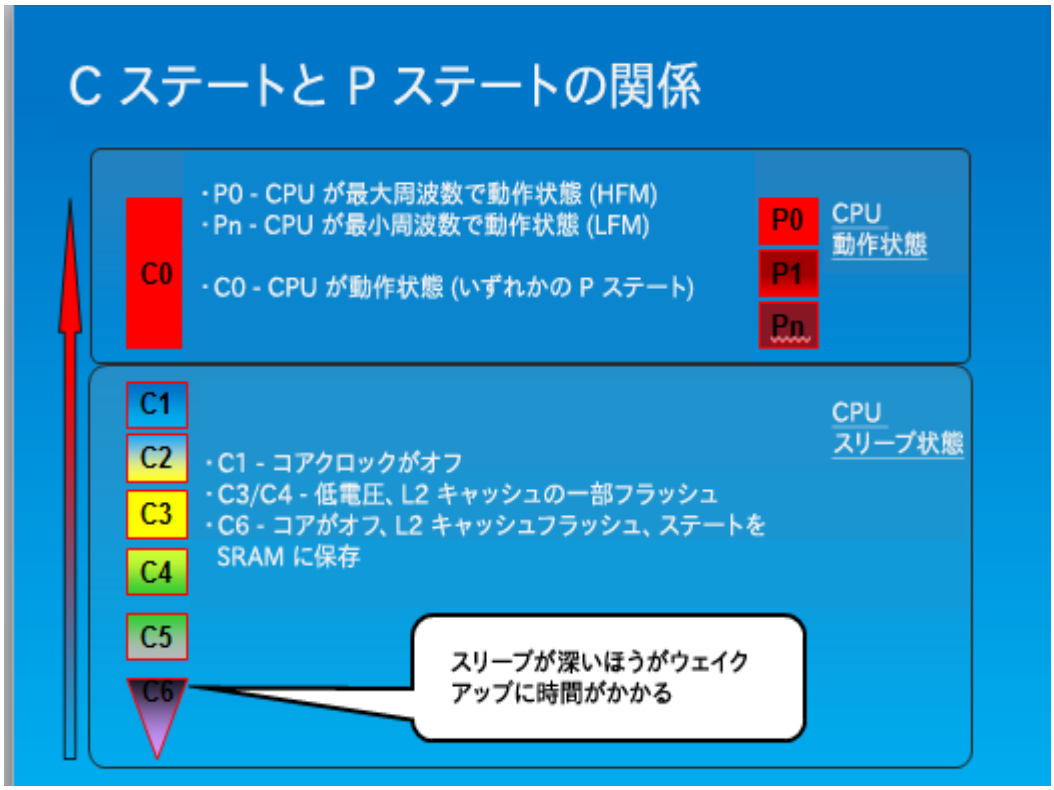
インテル® System Studio 2015 には、インテル® Energy Profiler と呼ばれる電力プロファイラーが含まれています。インテル® Energy Profiler により、スリープ状態、周波数/温度データを収集して、無駄な電力を消費しているソフトウェアを見つけることができます。ここでは、インテル® Energy Profiler の概要を説明します。

概要

システムの消費電力は、主に CPU スリープ状態と CPU 周波数から分かります。

- CPU スリープ状態 (C ステート)
  - CPU スリープ状態は、CPU がスリープ状態になる間隔を示します。システムがいつ、どのスリープ状態になり、なににによってウェイクアップしたかが分かります。スリープ状態が多いほうが、消費電力は低くなります。
- CPU 周波数 (P ステート)
  - CPU は、コアクロックの周波数を変えることもできます。周波数が高いほうが、消費電力が大きくなります。

次の図は、C ステートと P ステートの関係を示します。



ターゲットの Android* システムに応じて、異なるコレクターを使用する必要があります。

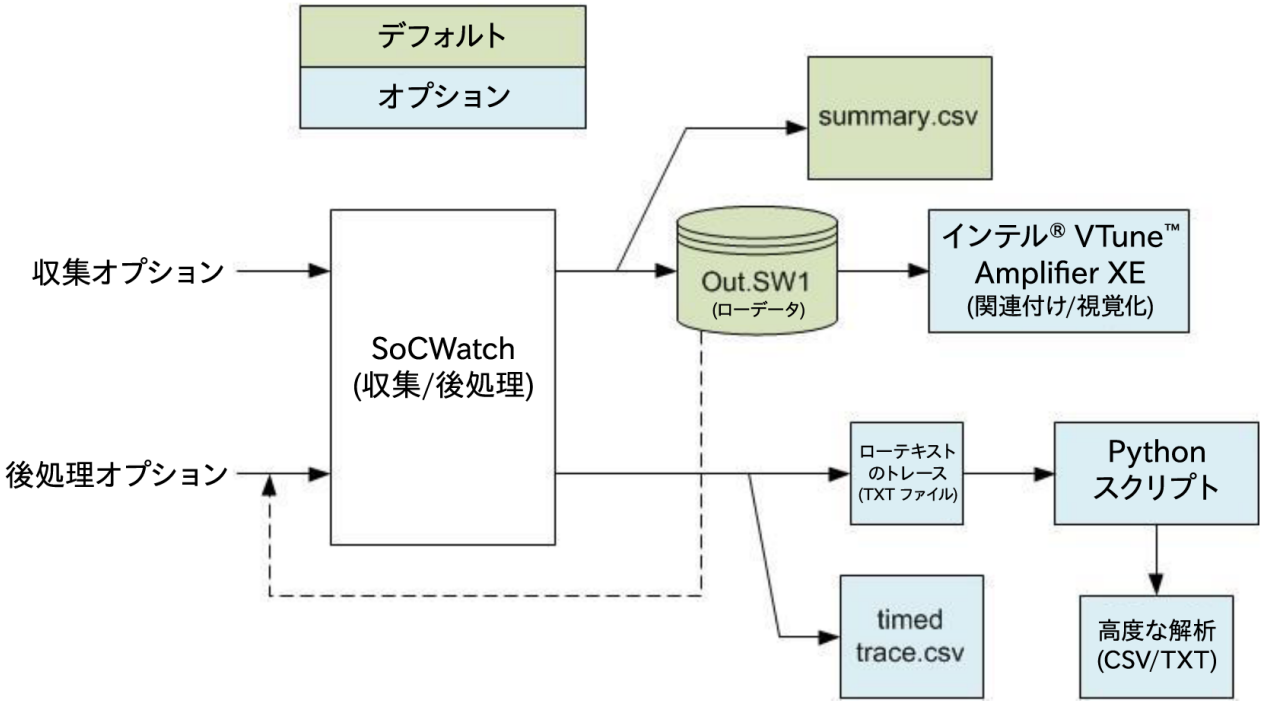
- Silvermont †/Haswell † システム
  - socwatch コレクター
- Clovertrail †+
  - wuwatch
- その他の組込み Linux* システム
  - amplxe-cl

ここからは、インテル® VTune™ Amplifier 2015 for Systems で socwatch を使用して、システムの C ステートと P ステートのデータを収集し、表示する方法を示します。

socwatch

SoC Watch (SoCWatch) は、インテル® アーキテクチャー・ベースのプラットフォームで消費電力に関連したシステム動作を監視するコマンドライン・ツールです。システムの電力効率に関する詳細を提供する、電力ステート、周波数、ウェイクアップ、その他の多種多様な測定基準を監視します。

以下の図に、socwatch の使用方法を示します。



インストール

現在、SoCWatch は Android* ベースのシステムをサポートしています。システム要件の詳細は、リリースノートを参照してください。

すでに Android* ベースのシステムに socwatch カーネルモジュールが統合されていることがあります。そうでない場合は、「カーネルモジュールをビルドする」セクションを参照してください。

インストール・スクリプトを実行する前に、adb によりホストとターゲットが接続されていることを確認してください。

1. ホストシステムで SoCWatch パッケージを展開後、Windows* ホストの Cygwin ウィンドウまたは Linux* ホストで次のファイル/スクリプトを実行します。Windows* ホストの場合: socwatch_android_install.bat ファイル

   Linux* ホストの場合: socwatch_android_install.sh スクリプト

   デフォルトでは、socwatch 実行ファイルがターゲットの /data/socwatch ディレクトリーにインストールされます。別のディレクトリーにインストールする場合は、-d オプションを指定します。adb により複数のデバイスがホストに接続されている場合は、–s オプションを指定して特定の Android* デバイスを指定できます。
2. adb コマンドを使用して、root 権限でシェルを開始します。

```
> adb root
> adb shell
```

3. ドライバーがあるディレクトリーに移動します。

   ```
   > cd /lib/modules
   ```
4. ドライバーをロードします。

   ```
   > insmod SOCWATCH1_*.ko
   ```
5. ドライバーがロードされたことを確認します。

   ```
   > lsmod
   ```
6. 必要に応じて、次のコマンドを実行してドライバーをアンロードします。

   ```
   > rmmod SOCWATCH1_*
   ```

カーネルモジュールのビルド

SoCWatch デバイスドライバーが OS イメージにインストールされていない場合は、ビルドする必要があります。

1. Android* をビルドするホストシステムに socwatch_android.zip ファイルをコピーします。
2. 次のコマンドを実行してファイルを展開します。

   ```
   > tar xvzf socwatch_android.zip
   ```

   socwatch_android ディレクトリーが作成されます。
3. cd コマンドを使用して socwatch_android/socwatch_driver ディレクトリーに移動します。
4. 次のコマンドを使用して build_driver スクリプトを実行します。

   ```
   > sh ./build_driver –k <kernel-build-dir>
   ```

   ここで、<kernel-build-dir> は、Android* カーネルのビルド中にローカルの Android* lib/modules ディレクトリーに置換されます。必要に応じて、–k オプションを指定します。build_driver スクリプトで適切な DEFAULT_KERNEL_BUILD_DIR 値が設定されている場合は、指定する必要はありません。

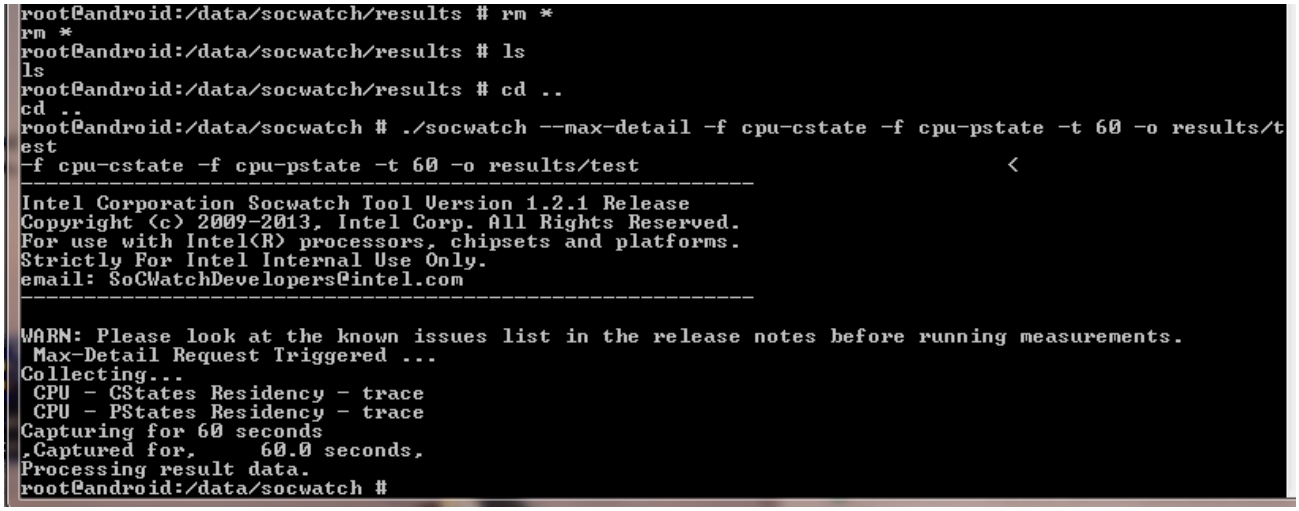
システムでデータを収集する

```
> cd /data/socwatch
> ../setup_socwatch_env.sh
```

システムで C ステートと P ステートのデータを収集するには、次の socwtach コマンドを実行します。

```
>./socwatch --max-detail –f cpu-cstate –f cpu-pstate –t 60 –
o ./results/test
```

このコマンドの実行には約 60 秒かかり、results ディレクトリーに test.sw1 と test.csv というファイルが生成されます。



収集したデータを表示するには、adb によりこれらのファイルをホストシステムにコピーする必要があります。

```
> adb pull /data/socwatch/results/test.csv c:\results
> adb pull /data/socwatch/results/test.sw1 c:\results
```

これらのファイルをインテル® VTune™ Amplfier 2015 for Systems にインポートするには、Windows* のプロンプトで次のコマンドを実行します。

1. 最初に、インテル® VTune™ Amplfier のインストール・ディレクトリーで amplxe-vars.bat ファイルを実行します。
2. amplxe-cl.exe -import test.sw1 C:\results\test を実行します。

C:\results に test という名前の結果ディレクトリーが作成されます。次のコマンドでこの結果ディレクトリーを開くことができます。

```
> amplxe-gui.exe C:\results\test
```

周波数のサマリー:

Elapsed Time: 60.358s

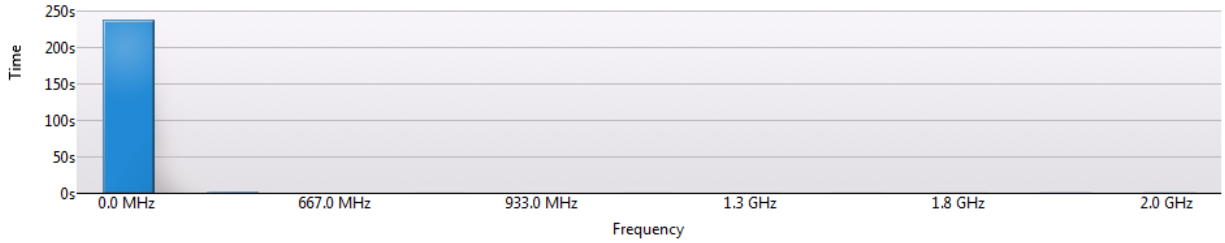
Available Core Time: 241.432s

Top 5 Frequencies

Frequency	Time (%)
0GHz	99.0%
0.533GHz	0.4%
1.99134GHz	0.1%
1.93061GHz	0.1%
1.78875GHz	0.1%
[Others]	0.2%

Frequency Histogram

This histogram represents a breakdown of the Elapsed Time per Frequency over all cores.



Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

Frequency:	1.5 GHz
Logical CPU Count:	4
Physical Core Count:	4
Operating System:	Linux
OS Detailed Name:	Android_4.2.2(kernel:3.4.43-188078-g01da887)
Computer Name:	localhost
Result Size:	1 MB

スリープ状態のサマリー:

Wake-ups/sec per Core: 21.824

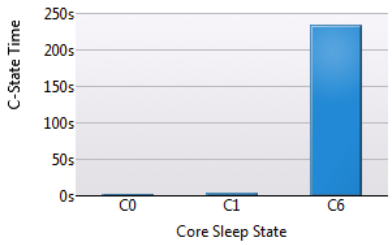
Elapsed Time:	60.358s
Available Core Time:	241.432s
CPU Utilization (%):	1.078
Total Time in Non-C0 States:	238.829s
Total Wake-up Count:	5,269
Wake-up Count due to IRQ Interrupts:	1,117
Wake-up Count due to Timers:	1,946

Top 5 Causes of Core Wake-ups

Wake-up Object	Wake-ups
IPI	29.8%
Kernel Timer	26.1%
IRQ 19 - mmc2	17.9%
User Timer	10.8%
Scheduler	9.0%
[Others]	0.0%

Elapsed Time per Core Sleep State Histogram

This histogram represents a breakdown of the Elapsed Time per Core Sleep State over all cores.



Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

Frequency:	1.5 GHz
Logical CPU Count:	4
Physical Core Count:	4
Operating System:	Linux
OS Detailed Name:	Android_4.2.2(kernel:3.4.43-188078-g01da887)
Computer Name:	localhost
Result Size:	1 MB

†開発コード名

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。