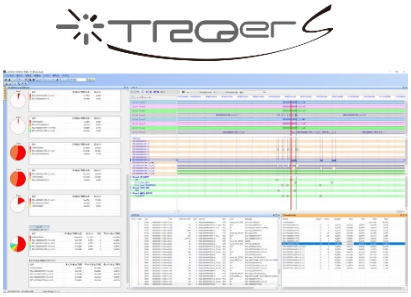
システムマクロトレースのご紹介

システムマクロトレースのご紹介



システムマクロトレース(以下、SMT)は、実際にユーザーシステム上で実行しているプログラムの実行履歴やタスク遷移を関数／スレッドチャートで表示することで、システム全体の動きを可視化することができる動的テストツールです。

システムの可視化のみならず、カバレッジ測定やスタック消費量／メモリ消費量の測定にも対応しております。

開発現場での困りごと

現在、産業(FA=Factory Automation)系での市場では、産業用ロボットやセンサーなど、様々な技術を実現するためにハイエンド SoC の採用が増え、より複雑化したソフトウェアやシステムの性能が求められています。

開発にあたり、以下のような困りごとはございませんでしょうか？

・プログラム上で問題を発見したが、何度やっても現象が再現しない。現象がまれにしか発生しない。

・print 文を入れてプログラムを実行すると、現象が再現しなくなる。ビルドに毎回時間がかかる。

・ソフトウェアが複雑化し、システム全体の挙動や流れを把握できない。

・マルチコア環境での処理を把握したい。各コアごとにどのようなタスクが行われているか確認したい。

・高負荷ソフトウェアのボトルネック/ホットスポット箇所を特定したい。

特長

・**プログラムの動きが一目瞭然！マルチコア環境にも対応！**

実動作でのプログラムの実行履歴やタスク遷移を可視化することで、システムの動きを容易に把握することができます。
また、マルチコア環境に対しても、各コアごとに実行履歴やタスク遷移を確認することができます。

・**バグが簡単に探せ、短時間で効率よく解決！**

まれにしか発生しない不具合現象も、動的解析／動的テストを行うことで効率よく解決することができます。

・**テスト／デバッグなど開発工数を大幅に削減！**

単体テストから結合テストまで、目的やレベルに合わせて幅広く使用できるため、テスト工数やデバッグ工数の削減を実現します。

・**システムのパフォーマンスを分析し、ボトルネックとなっている処理を把握！**

複雑化したソフトウェアに対しパフォーマンス分析や時間計測を行うことで、ボトルネックとなっている処理を特定できます。

対応 OS

Linux、RTOS、non-OS など様々なプラットフォームで動作するアプリケーションに対応可能です。

ユースケース例

「プラットフォームの置き換えで、ハードウェアをスペックアップしたものの、実機動作で目標スペックに届かない。」といった課題に対して、実際に SMT と Raspberry Pi を利用したシステムの処理性能や時間計測のユースケースをご紹介します。

Raspberry Pi 2B → Raspberry Pi 3B+ へ置き換えた際に、どれだけスペックが向上するかを数値化する例となります。

※OS、アプリケーション、ドライバなどは同条件で実行

	Raspberry Pi2B	Raspberry Pi3B+
CPUコア	Cortex-A7×4	Cortex-A53×4
CPU周波数	900MHz	1.4GHz
CPUバス幅	32bit	64bit

以下は、ドライバを呼び出すテストアプリを shell スクリプトで 5 回連続実行した際の実行結果です。

関数チャート(Raspberry Pi 2B)

関数チャート(Raspberry Pi 3B+)

テストアプリの開始地点～次の開始地点にマーカーを付け、その実行間隔(赤線～緑線)を計測したところ、

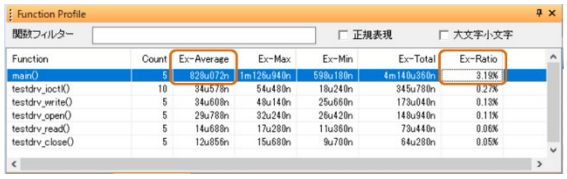
・Raspberry Pi 2B
　アプリの実行間隔 ：7m 486u 600n sec

・Raspberry Pi 3B+
　アプリの実行間隔 ：5m 528u 940n sec

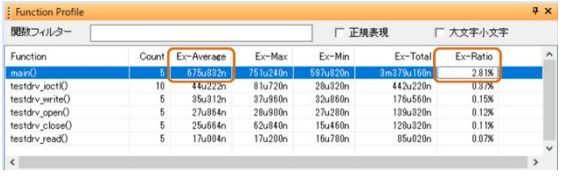
となり、Raspberry Pi 2B → Raspberry Pi 3B+で約 26%の時間短縮になっていることが分かります。

また、以下はテストアプリとドライバの実行回数や CPU 使用率の割合を分析した結果となります。

関数プロファイル(Raspberry Pi 2B)



関数プロファイル(Raspberry Pi 3B+)



テストアプリの平均実行時間と CPU 使用率は、

・Raspberry Pi 2B
　平均実行時間 ：828u 072n sec, CPU 使用率 ：3.19%

・Raspberry Pi 3B+
　平均実行時間 ：675u 832n sec, CPU 使用率 ：2.81%

となり、Raspberry Pi 3B+の方が、テストアプリの平均実行時間やシステム全体の動作におけるアプリ実行時間の割合が少ないことが分かります。

次に、CPU 使用率のチャート表示を並べて確認してみます。

CPU 使用率（Raspberry Pi 2B）

CPU 使用率（Raspberry Pi 3B+）

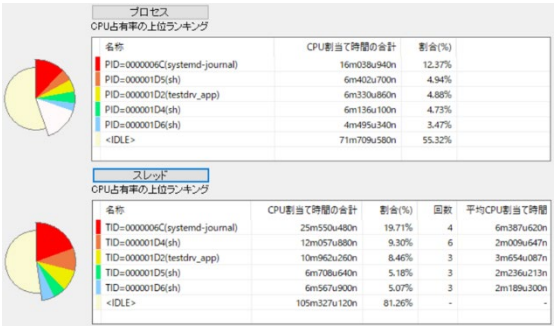
Raspberry Pi 2B が平均 50%程度の稼働に対し、Raspberry Pi 3B+は所々100%に達しており、平均 60%程度と CPU の負荷が高いことが分かります。

通常は、置き換え後の機種(今回の例では Raspberry Pi 3B+)の方がスペックが向上しているため、CPU 使用率についても負荷が少なくなるものと思われます。

なぜ、Raspberry Pi 3B+の方が余力なく見えるのでしょうか？

以下は、各コアの総合的な CPU 占有率を分析した結果となります。

プロセス／スレッド一覧（Raspberry Pi 2B）



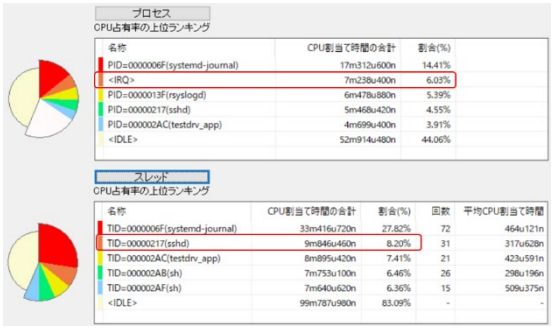
プロセス
CPU占有率の上位ランキング

名称	CPU割当て時間の合計	割合(%)
PID=0000006C(systemd-journal)	16m038u940n	12.37%
PID=000001D5(sh)	6m402u700n	4.94%
PID=000001D2(testdrv_app)	6m330u860n	4.88%
PID=000001D4(sh)	6m136u100n	4.73%
PID=000001D6(sh)	4m495u340n	3.47%
<IDLE>	71m709u580n	55.32%

スレッド
CPU占有率の上位ランキング

名称	CPU割当て時間の合計	割合(%)	回数	平均CPU割当て時間
TID=0000006C(systemd-journal)	25m550u480n	19.71%	4	6m387u620n
TID=000001D4(sh)	12m057u880n	9.30%	6	2m009u647n
TID=000001D2(testdrv_app)	10m962u260n	8.46%	3	3m654u087n
TID=000001D5(sh)	6m708u640n	5.18%	3	2m236u213n
TID=000001D6(sh)	6m567u900n	5.07%	3	2m189u300n
<IDLE>	105m327u120n	81.26%	-	-

プロセス／スレッド一覧（Raspberry Pi 3B+）



プロセス
CPU占有率の上位ランキング

名称	CPU割当て時間の合計	割合(%)
PID=0000006F(systemd-journal)	17m312u600n	14.41%
<IRQ>	7m238u400n	6.03%
PID=0000013F(rsyslogd)	6m478u880n	5.39%
PID=00000217(sshd)	5m468u420n	4.55%
PID=000002AC(testdrv_app)	4m699u400n	3.91%
<IDLE>	52m914u480n	44.06%

スレッド
CPU占有率の上位ランキング

名称	CPU割当て時間の合計	割合(%)	回数	平均CPU割当て時間
TID=0000006F(systemd-journal)	33m416u720n	27.82%	72	464u121n
TID=00000217(sshd)	9m846u460n	8.20%	31	317u628n
TID=000002AC(testdrv_app)	8m895u420n	7.41%	21	423u591n
TID=000002AB(sh)	7m753u100n	6.46%	26	298u196n
TID=000002AF(sh)	7m640u620n	6.36%	15	509u375n
<IDLE>	99m787u980n	83.09%	-	-

プロセス一覧を確認すると、Raspberry Pi 2B では動いていない sshd スレッドが Raspberry Pi 3B+で動作しており、CPU 占有率が 2 位になっています。これにより割り込み(IRQ)のプロセスも動作し、さらに CPU 占有率が上がっていることが分かります。

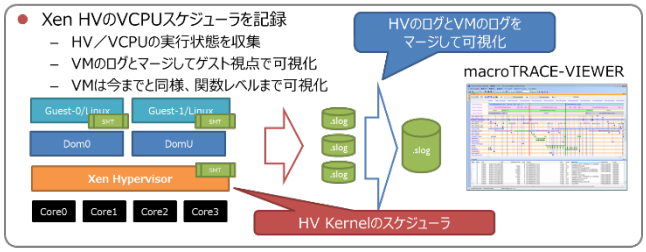
つまり、アプリ実行時に Raspberry Pi 3B+で ssh 通信を止め忘れていたことが原因で、CPU の稼働率が上がっていたことが分かります。

このように、ハードウェアをスペックアップしたにも関わらず、実機動作では目標スペックに届かないといった場合でも、動作している関数やプロセス／スレッドの中身を分析することで、原因を特定することが可能になります。

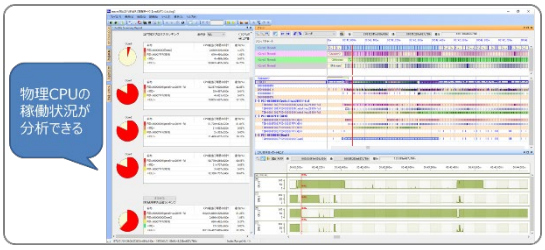
Hypervisor 対応

近年では Hypervisor 搭載のシステムも増加しており、システムマクロトレースは、Hypervisor にも対応しております。
以下は Xen Hypervisor におけるコンテキストや VCPU スケジュールの可視化を行った例となります。

HV と VM のコンテキストを可視化



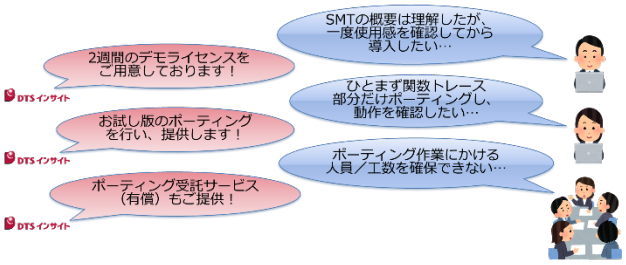
Xen と VCPU のスケジュール分析



SMT、一度使ってみませんか？

SMT をお試しで使用してみて、使用感を確認してみたい！
というお声も多いかと思います。

弊社では、ツールのデモライセンスを 2 週間分ご用意しておりますので、SMT を実際にお試しで使用していただくことができます。



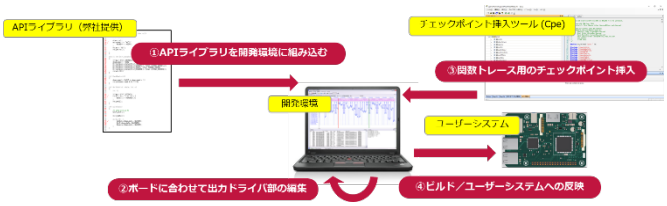
また、まずは関数トレース部分だけポーティングして動作を確認したいという場合でも、関数トレースのみのポーティングを弊社にて行い、お試し版としてご提供も可能です。

また、ポーティング作業にかける人員や工数を確保できない、という場合でも、以下の SMT ポーティング受託サービスをご用意しておりますので、ぜひともご活用いただければ幸いです。

SMT ポーティング受託サービスのご紹介

弊社では、お客様の SMT のご導入をさらに容易にするため、API ライブラリのポーティング受託サービスを行っております。

ポーティング受託サービスでは、ユーザ環境への API ライブラリ組み込みから動作確認／評価、ポーティング手順書の作成まで全て弊社にて対応致します。



お問い合わせ先

ご興味がございましたら、以下の宛先までメールにてお問い合わせください。
別途、弊社担当より、詳細についてご案内致します。
その他、デバッグ作業についてもお困りごとがございましたら、些細な事でも構いませんのでお気軽にご相談ください。

株式会社ＤＴＳインサイト
プロダクトソリューション事業部 営業部 営業技術課
E-mail：support@dts-insight.co.jp