Vol.2016-HPC-153 No.18
2016/3/2

# プロファイラにおける汎用的な CPU 性能情報と表示機能

阿部文武† 中村朋健† 志田直之†

**概要**：アプリケーションの性能解析に用いるプロファイラにおいて，汎用的に利用できる CPU 性能情報を出力する機能を開発する．スーパーコンピュータ「京」や PRIMEHPC FX10/FX100 のプロファイラは，プロセッサのハードウェアカウンタから取得できる情報をプロプライエタリな形式で取り扱い，Microsoft Excel で可視化する形式で実現されている．そのため，必要なデータだけを取り出すといった情報の取り扱いやすさの点，および情報表示のために利用できるプラットフォームが制限されているという使い勝手の点で課題を抱えている．本稿では，プロファイラが内部的に取り扱うデータを独自形式から XML に変換する仕組みを構築し，可読性および汎用性を高くする方法を提案する．この方法を用いることよって，利用者は必要な情報だけを取得したり，データを可視化する環境を自由に選択したりするなどのメリットが期待される．

## 1. はじめに

システムの利用時間や利用資源を制限される HPC システムにおいて，実行するアプリケーションの性能のチューニングは重要である．チューニングにはプロファイラやトレーサなどの性能解析ツールが用いられる．

性能解析ツールの代表的なものとして，プロファイリングやトレーシングの統合的なツールである Score-P [1]や TAU [2]，バイナリアプリケーションの性能解析を行う MAQAO [3]，トレーシング情報を可視化する Vampir [4]などが挙げられる．

スーパーコンピュータ「京」[5]（以下,「京」）や, FUJITSU Supercomputer PRIMEHPC FX10 [6]（以下，FX10）および PRIMEHPC FX100 [7]（以下，FX100）には，基本プロファイラおよび詳細プロファイラと呼ばれる独自のプロファイラが 2 つ実装されている[8]. 基本プロファイラはサンプリングにより一定間隔で走行情報を取得することで，測定対象のプログラムを再翻訳することなく，性能特性の傾向を解析可能である．詳細プロファイラは基本プロファイラが解析した性能が低い箇所を厳密に調査するために用いられる．詳細プロファイラはアプリケーション中の特定のプログラム箇所に対して，サービスライブラリを呼び出す形式で実現する．具体的には，アプリケーションのプログラムコード内に測定区間の開始および終了処理コードを挿入し，翻訳したアプリケーションを実行することで，測定区間における処理の呼び出し回数や実行時間などを測定することができる．

詳細プロファイラは機能の一部として精密 PA 機能を持っている．精密 PA 機能は SPARC64 プロセッサのハードウェアカウンタを用いて, 命令実行, メモリアクセス待ち, 演算待ちなどの CPU の動作状態ごとの実行サイクル数をカウントすることができる．この結果を利用して，CPU 内のどの処理にボトルネックがあるのかを分析することが可能になる．この機能はサイクルアカウンティング分析[9]と呼ばれる．

このように，精密 PA 機能は性能改善に有効な情報を数多く収集することができる点で，他のプロファイラと比較しても優位性を持った機能である．一方で，図 1 に示すように，精密 PA 機能が情報を取り扱う仕組みはブラックボックス化しており，利用者は取り扱われている情報を加工することは難しく, 汎用性の点で改善の余地がある. また，精密 PA 機能は利用できるシステムが「京」や PRIMEHPC に限られているため，情報を加工する仕組みを作ったとしても，他のシステムに転用することはできない点も汎用的に利用できない原因となる．

そこで, 本稿ではプロファイラの情報体系を整理し XML 形式で取り扱うようにした手順と，開発した既存の情報を XML に変換する仕組みについて説明する．2 章では，現行の「京」，FX10 および FX100 において CPU 性能情報を取得する仕組みやその課題について説明する．3 章では，課題に対する解決案を提案し，プロファイル情報を体系的にまとめた手順について説明する. 4 章では，精密 PA 機能の汎用性をさらに向上させ，より多くのシステムに展開するために開発した情報加工スクリプト（XMLconverter）に関する説明とその評価を行う．5 章では，今回提案した汎用性の向上への取り組みで不足している点や改善する点を挙げ，今後対応すべき項目をまとめる．

† 富士通株式会社 次世代テクニカルコンピューティング開発本部
Fujitsu Limited, Next Generation Technical Computing Unit

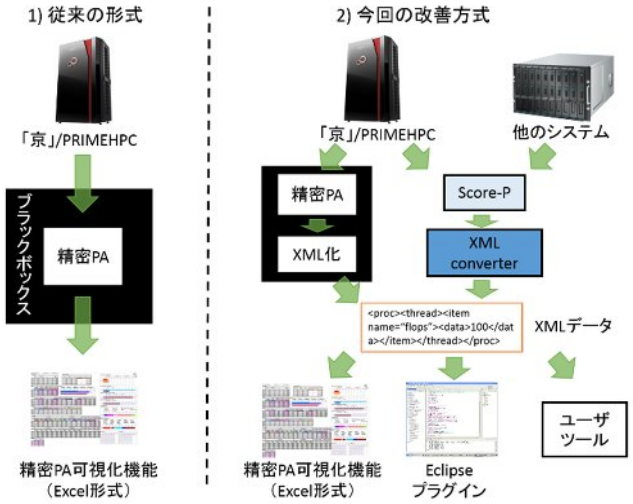


図 1　本研究で目指す改善方式

## 2. CPU 性能情報における課題

### 2.1 精密 PA 機能

詳細プロファイラが CPU 処理の実行回数などを厳密にカウントできるのは，CPU のハードウェアカウンタを利用しているためである．このハードウェアカウンタを利用して CPU の性能解析を可能にする仕組みは，精密 PA（Performance Analysis）機能と呼ばれている．

精密 PA 機能で取得できる主な情報を表 1 から表 6 にまとめる． 表 1 はパフォーマンス情報である．これは，測定対象のアプリケーションの実行時間や実行性能を示している．表 2 はメモリやキャッシュのスループットに関する情報である．これは，メモリやキャッシュ間のデータ転送効率や稼働率を示している．表 3 は SIMD 命令に関する情報である．これは，SIMD によって命令がどのくらい並列化されたかを示している．表 4 はキャッシュミス情報である．これは，キャッシュがどのくらい効率的に利用されたかを示している．表 5 は命令数に関する情報である．これは，各命令の種類がどのような割合で実行されたかを示している．表 6 はロードバランス情報である．これは，スレッド毎の処理のバランスを示している．

表 1　パフォーマンス情報

| 出力項目 | 出力項目の意味 |
|---|---|
| 実行時間 | スレッド毎の待ち時間と演算数の和 |
| 浮動小数点演算ピーク比 | 1 コア当たりの論理ピーク性能値に対する実行した浮動小数点演算命令数 |
| MFLOPS | 実行時間に対する実行した浮動小数点演算数 |
| MIPS | 全実行時間に対する総実効命令数 |
| 整数演算性能（MINOPS） | 実行時間に対する実行した整数演算数 |
| 浮動小数点演算数 | 浮動小数点演算の総数 |
| 整数演算数 | 整数演算の総数 |

表 2　メモリ・キャッシュスループット情報

| 出力項目 | 出力項目の意味 |
|---|---|
| メモリスループット | 全実行時間に対する 2 次キャッシュとメモリ間の転送量 |
| L2 スループット | 全実行時間に対する 1 次データキャッシュと 2 次キャッシュ間の転送量 |
| メモリビジー率 | メモリの稼働率（2 次キャッシュとメモリ間の転送処理の稼働率） |
| L2 ビジー率 | 2 次キャッシュの稼働率（L1D⇔L2 間の転送処理と L2⇔メモリ間の転送処理を制御する装置の稼働率） |
| L1 ビジー率 | 1 次キャッシュとレジスタ間のパイプライン稼働率 |

表 3　SIMD 命令情報

| 出力項目 | 出力項目の意味 |
|---|---|
| SIMD 命令率 | 総実効命令数に対する SIMD 命令数の割合 |
| SIMD 浮動小数点演算命令率 | 浮動小数点演算命令数に対する SIMD 浮動小数点演算命令数の割合 |
| SIMD 整数演算命令率 | 整数演算命令数に対する SIMD 整数演算命令数の割合 |
| SIMD ロード・ストア命令率 | 総ロード・ストア命令数に対する SIMD 化されたロード・ストア命令数の割合 |

表 4　キャッシュミス情報

| 出力項目 | 出力項目の意味 |
|---|---|
| L1 ミス率 | 総実効命令数に対する 1 次命令キャッシュミス回数の割合 |
| L1D ミス率 | ロード・ストア数に対する 1 次データキャッシュミス回数の割合 |
| ロード・ストア数 | ロード・ストア命令の総数 |
| L1D ミス数 | 1 次データキャッシュミス回数 |
| L2 ミス率 | ロード・ストア数のうち 2 次キャッシュミス回数の割合 |
| L2 ミス数 | 2 次キャッシュミス回数 |

表 5　命令数情報

| 出力項目 | 出力項目の意味 |
|---|---|
| ロード・ストア命令率 | 総実効命令数に対するロード・ストア命令数の割合 |
| 浮動小数点演算命令率 | 総実効命令数に対する浮動小数点演算命令数の割合 |
| 整数演算命令率 | 総実効命令数に対する整数演算命令数の割合 |
| プリフェッチ命令率 | 総実効命令数に対するプリフェッチ命令数の割合 |
| 分岐命令率 | 総実効命令数に対する分岐命令数の割合 |
| パーミュテーション命令率 | 総実効命令数に対するパーミュテーション命令（レジスタの中で順序を入れ替える）数の割合 |
| コンカチシフト命令率 | 総実効命令数に対するコンカチシフト命令（倍精度浮動小数点レジスタの有効な全要素を連結して左シフトを行う）数の割合 |

表 6　ロードバランス情報

| 出力項目 | 出力項目の意味 |
|---|---|
| ロードバランス | プロセス内の全スレッドで最も長い実行時間に対する各スレッドの実行時間の割合 |

| | |
|---|---|
| 命令バランス | プロセス内の全スレッドで最も多い総実効命令数に対する各スレッドの総実効命令数の割合 |

これら取得されたデータは，以下の図 2 で示すように Microsoft Excel 上でマクロを使用することで可視化され，利用者が情報を確認できる．図 2 の中ほどにある縦の棒グラフがサイクルアカウンティング分析のグラフである．サイクルアカウンティングは各命令コミットや待ち状態に費やした CPU サイクル数を項目ごとに積み上げたものである．図 3 に示すように，CPU サイクルに費やした項目が色の違いで表現されている．

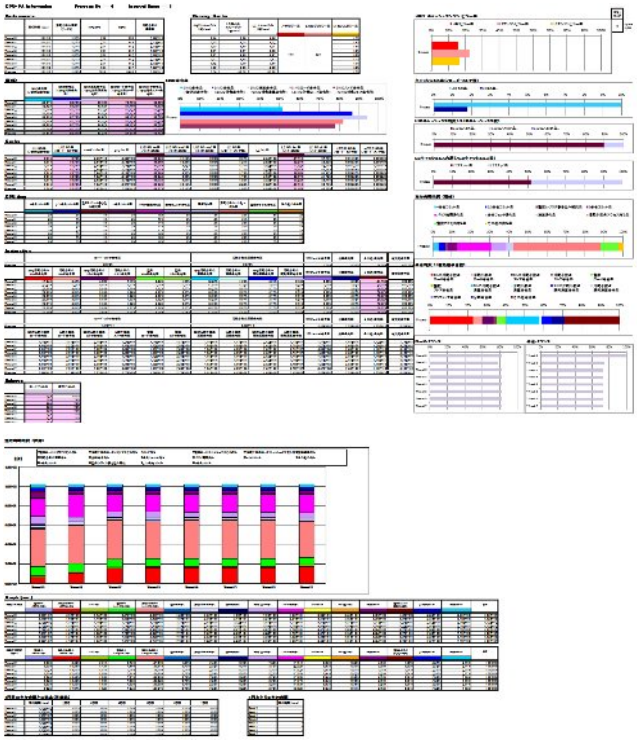

図 2　精密 PA 可視化機能（Excel 形式）


図 3　サイクルアカウンティング分析

### 2.2 精密 PA 機能の課題

精密 PA 機能には大きく 2 つの課題がある．一つは出力が Microsoft Excel に固定されている点であり，もう一つはプロファイルデータを取得する手段が詳細プロファイラであり「京」や FX10 および FX100 といったマシンに限られている点である．

#### 2.2.1 出力における課題

精密 PA 機能の出力が Microsoft Excel に固定されることにより，以下の問題が存在する．

- 解析・実行環境制限の問題
  - 結果の確認処理は Windows を中心としたクライアントマシンに限られる
- 情報再利用の問題
  - 加工された情報がグラフなどで表示されるため，利用者が必要な情報を抜き出して加工したり再利用したりしにくい

解析・実行環境制限の問題は，ユーザにとって負荷を強いている問題である．たとえば，ジョブの実行などはターミナルから ssh 接続を使用してコマンドラインでプログラムを作成し，ジョブを実行し，結果を確認するといった一連の流れがあるのに対して，精密 PA 情報だけはクライアントマシンに性能結果ファイルを転送し，Microsoft Excel を立ち上げ内容を確認するというわずらわしさが発生する．一部の情報だけを確認したいときに，オーバーヘッドが大きくなる．

情報再利用の問題は，ユーザにとって利便性を向上させる点での問題である．2.1 節で各種情報を示したように精密 PA 機能が取得するデータは多岐にわたる．これらのデータを Excel ファイルに加工した形でユーザに提供しているが，ユーザが性能分析に一部の情報しか利用しないことがある．このとき，ユーザは自身にとって必要な精密 PA 情報だけを取捨選択し，情報を加工することができれば，ユーザにとってより扱いやすいシステムになる．

#### 2.2.2 プロファイルデータ生成における課題

詳細プロファイラに実装されている精密 PA 機能が利用できる環境は「京」や FX10 および FX100 に制限されている．仮に，利用者が詳細プロファイラに実装されている精密 PA 機能で採取できた情報を加工できるような仕組みを作ったとしても，他のシステムに転用することができないため，汎用性が低いことになる．

## 3. 精密 PA 機能の解決案

### 3.1 出力における解決案

精密 PA 機能の出力における課題は，Excel 出力される処理がブラックボックス化されており，情報の作られ方がユーザにとってわからない点が一因と言える．

そこで，精密 PA 機能が内部的に取り扱っているデータの可読性を高めて，Excel 出力の前にワンクッション置く形で情報出力する方式を検討した．精密 PA 情報の可読性を高めることで，出力時の問題に対して以下の効果が期待できる．

- 解析・実行環境制限の問題
  - サーバ側での情報加工が可能になり，情報転送が不要になる
- 情報再利用の問題
  - 情報の抽出および加工が可能になり，必要な情報だけ選択できる

可読性を高めるために, 精密 PA 情報を XML で表現することを選択した．精密 PA 情報を利用する側がデータを取り扱えるようにするには, XML の記述ルールを定義する必要がある．そのためには，まずデータ構造を定義する．精密 PA 機能で取得したデータはプロセス毎，スレッド毎に保持されるため，データ構造の大きな枠組みとしてプロセスを定義し，その中にスレッドの枠組みを用意する．そしてスレッドの中に，精密 PA 機能で取得できる各情報を定義する．定義したデータ構造を図 4 に示す．このデータ構造で定義された以下の要素を, XML の要素として定義する．

- プロセス: <process id=””>
- スレッド: <thread id=””>
- 出力項目名: <item name=””>
- 出力項目の実データ: <data unit=””>

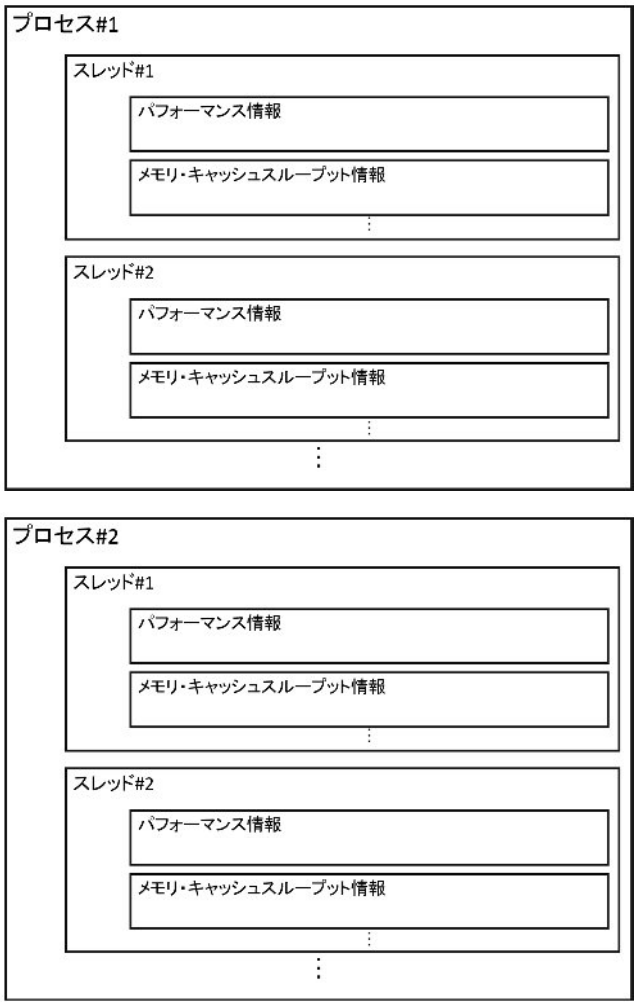


図 4　精密 PA 情報のデータ構造化

プロセスにはプロセス番号，スレッドにはスレッド番号といった他のデータとの識別要素があるため，これを属性 id に格納する．精密 PA 機能で取得したハードウェアカウンタの値は，data タグに挟んで格納する．data が示す値の単位は属性 unit に格納する．これらのタグ，属性を使って表現した XML 文書の例を図 5 に示す．図 5 の XML 表記は，プロセス番号 1 のプロセスが 1 つあり，その中にスレッド番号 0 のスレッドが 1 つ含まれており，このスレッドの実行時間(elapsed_time)が 600 秒であることを示す例である．

```
<process id=”1”>
  <thread id=”0”>
    <item name=”elapsed_time”>
      <data unit=”sec”>600</data>
    </item>
  </thread>
</process>
```

図 5　精密 PA 情報の XML 表記

### 3.2 プロファイルデータ生成における解決案

プロファイルデータを生成する基盤を共通化することで，「京」，FX10 および FX100 以外のより多くのシステムに適用することができないか検討した．具体的には，精密 PA 情報の取得に，PC クラスタ環境やさまざまな HPC システムで利用可能であるオープンソースソフトウェアの Score-P を選択した．Score-P は図 6 に示すように，プロセス並列やスレッド並列に関する性能情報などを計測用のラッパを用いてシステム間の差異を吸収し，プロファイラである TAU やトレーサである Vampir などへの適用が可能となる基盤である．情報計測にはハードウェアカウンタを利用することも考慮されていることから，精密 PA 機能と同等の情報を取得することが可能である．

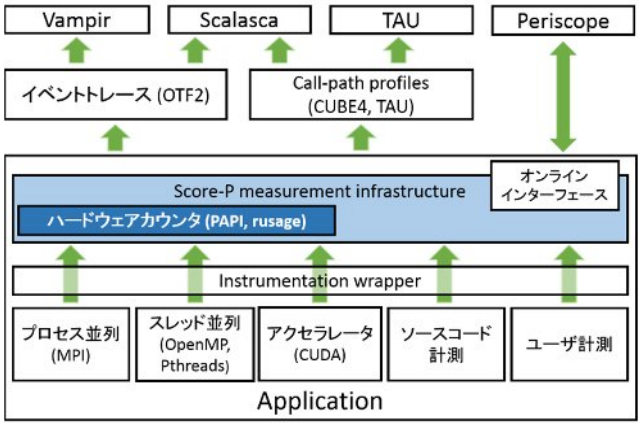


図 6　Score-P のアーキテクチャ

Score-P で取得したプロファイルデータを精密 PA 機能の情報として扱うには，データを 3.1 節で定義した XML 表記に加工する必要がある．Score-P で取得したプロファイルデータ（Profile.cubex）を精密 PA 情報（XML ファイル）に変更する際に，Cube [10]の CUI である cube_info コマンドと cube_dump コマンドを利用して必要な情報を抽出した．cube_info コマンドはプロファイルデータの様々な情報を表示するコマンド，cube_dump コマンドはプロファイルデータの一部を表示するコマンドである．Score-P で抽出したプロファイルデータから精密 PA 情報へ変換するまでのフローを図 7 に示す．図 7 の変換処理を XMLconverter と呼ぶこととする．XMLconverter は Python を用いて実装した．Python を利用したのは，XMLconverter は文字列の解析や加工を行うので，スクリプト言語が取り扱いやすいと判断したためである．

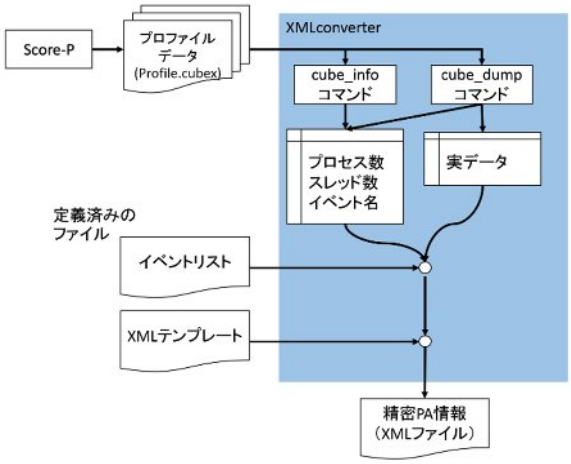


図 7　XMLconverter の処理フロー概要

cube_info と cube_dump それぞれ単体のコマンドではプロセスとその中に含まれるスレッドの関係を明確にすることができない．そこで，それぞれのコマンドの実行時のオプションを変えながら複数回呼び出すことによって，プロセスとスレッドの階層関係を含む精密 PA 情報を実現した．さらにこのとき，スレッド内にはハードウェアカウンタで取得する個々の項目（イベントリスト）を取得する．プロセス，スレッドおよびイベントリストは，事前に定義しておいた XML のテンプレートと照らし合わせて XML 形式に加工され，精密 PA 情報（XML ファイル）として出力される．実際に XMLconverter で生成した XML ファイルの例を図 8 に示す．

```
<process id="0">
 <thread id="0">
  <category name="instruction1">
   <item name="instruction_counts">
    <data unit="counts"/>
    <item name="effective_instruction_counts">
     <data unit="counts">
      44905883908
     </data>
     <item name="SIMD_instructions">
      <data unit="counts"/>
      <item name="XSIMD_load_store_instructions">
       <data unit="counts">
        6251121515
       </data>
       <item name="SIMD_load_store_instructions">
        <data unit="counts">
         4951441337
        </data>
```

```
<item name="SIMD_fl_load_instructions">
 <data unit="counts">
  4123285617
 </data>
</item>
……
```

図 8　Score-P のプロファイルデータから生成した精密 PA 情報（XML ファイル）の出力例

## 4. XMLconverter の評価

今回開発した XMLconverter による精密 PA 情報（XML ファイル）の処理時間と精密 PA 情報のファイルサイズについて評価する．性能データの取得対象としたプログラムと，XMLconverter を動作させた実行環境は以下のとおりである．

- プログラム：NAS Parallel Benchmarks [11], CG, Class=B（プロファイルデータの生成は FX100 で実行）
- 実行環境
  - サーバ: PRIMERGY RX200 S8
  - CPU: インテル® Xeon® プロセッサ E5-2630v2, 2.6GHz, 6 コア/12 スレッド
  - メモリ: 32GByte

### 4.1 処理時間の評価

XMLconverter の処理時間について評価する． 測定対象となるプログラムのプロセス数によって出力項目数が変化するため，プロセス数の違いによる処理時間を比較した．比較結果を表 7 および図 9 に示す．ここで，real とはプログラムの実時間，user とはユーザ CPU 時間，sys とはシステム CPU 時間を示す．

表 7　プロセス数による XML 化処理時間の違い

| Proc 数 / [sec] | 4 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|
| Real | 6.68 | 7.82 | 11.97 | 28.06 | 86.82 | 308.43 |
| User | 3.20 | 4.74 | 8.87 | 24.67 | 83.05 | 304.03 |
| Sys | 2.63 | 2.34 | 2.40 | 2.60 | 2.93 | 3.44 |

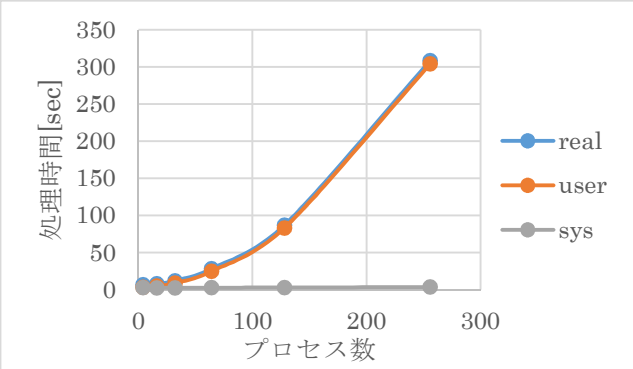


図 9　プロセス数による XML 化処理時間の比較

図 9 の結果から以下のことが明確となった．

- プロセス数に対して処理時間はほぼ線形に増加
- 処理時間はユーザ CPU 時間が支配的であり，XMLconverter の作りに起因する

1 プロセス当たりの処理時間が 1 秒以上かかっているため，XMLconverter の処理論理に問題があることは明確である．そこで，XMLconverter を見直し，ボトルネックを絞り込んだところ，図 7 で示される XML テンプレートに定義されている項目と実際に採取されたイベント名を検索しマッチングさせる処理に原因があることがわかった．実際の処理の 9 割の時間がこのマッチングに費やされていた．

### 4.2 データサイズの評価

精密 PA 情報（XML ファイル）のデータサイズについて評価する．XMLconverter によって生成された XML ファイルのサイズを，測定対象プログラムのプロセス数の違いによって比較した（表 8，図 10）．

表 8　プロセス数による XML ファイルサイズの違い

| プロセス数 | ファイルサイズ[kbyte] |
|---|---|
| 16 | 865.8 |
| 32 | 1731.3 |
| 64 | 3463.1 |
| 128 | 6923.4 |
| 256 | 13845.8 |

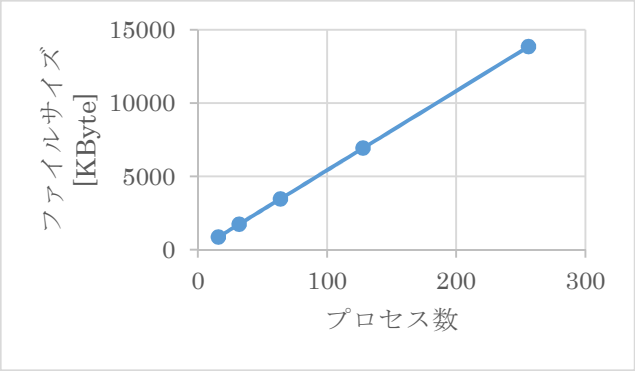


図 10　プロセス数による XML ファイルサイズの比較

1 プロセスでも 50k バイトを超えるファイルサイズであり，ファイルサイズはプロセス数に対してほぼ線形に増加している．仮に，「京」で利用可能なノードすべてに並列プロセスを割り当てるとすると，82,944 プロセス並列のジョブが実行できる．このときのファイルサイズは 4.3G バイトとなり非常に大きなファイルサイズになる．

今回作成した XMLconverter は，人による可読性を高めるために XML のタグのレベルに応じてインデントを追加していることがファイルサイズの大きさの一因である．インデントを省略することで，データサイズは 60%削減できた．

## 5. 今後の課題

XML ファイルへの加工については前章の評価で示したように以下の課題が明確になっている．

### 5.1 データ加工の問題

XMLconverter では，ファイルサイズの問題および加工時間の問題が明確になった．精密 PA 情報のファイルサイズの問題については，人による可読性を高めるために追加したインデントがファイルサイズの大きくした一因である．実際にインデントを無くすことで，データサイズは 60%削減することができた．60%減ったとしても高並列時のデータサイズは大きいため，さらなる解決策が求められる．そこで，以下の対処方法を検討している．

- 精密 PA 情報（XML ファイル）を圧縮する
- YAML [12]などの XML 以外のデータ形式を採用する

データ圧縮に関しては，出力されたファイルを ZIP で圧縮したり，EXI フォーマット[13]にしたりする方法などが考えられる．可読性は維持したままこれらの対応が可能であるか，今後，設計や実装し，新たに検証していきたい．

XMLconverter の処理時間の問題については，XMLconverter の実装方法が原因である．改善方法として，データ構造や検索処理のアルゴリズムの見直しや，今回使用した Python 以外のスクリプトや C 言語のプログラムに変更することが考えられる．こちらについても，設計を見直して検討を進めていきたい．

### 5.2 データ出力の問題

精密 PA 情報（XML ファイル）を生成する処理を実装，評価したが，そのファイルを実際に活用できるかたちにするのは今後の課題である．

XML のパーサを使えば，比較的容易に好きな情報だけを抜き出して，表やグラフを独自に作成することも可能となる．たとえば，図 11 は XML から抽出した命令コミット（1 マシンサイクルで N 命令実行した時間）の情報をサイクルアカウンティングのグラフとして加工したものである．ここでは，4 つのプロセスについて 1 から 4 命令コミットの時間と命令がストールしている時間（0 命令コミット）を積み上げ，グラフ全体の高さがプロセス実行の総時間を示している．この例では，表示化のために JavaScript と HTML5 を利用している．

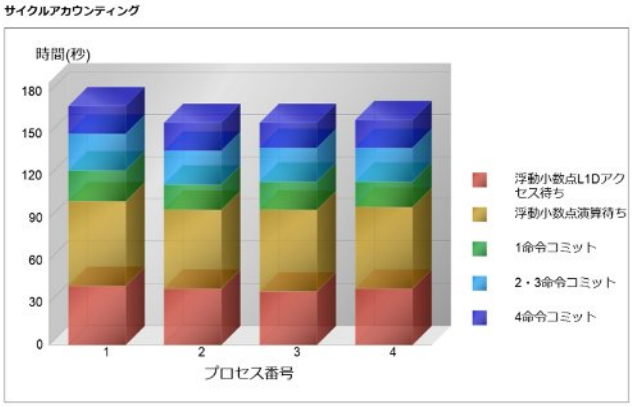


図 11　サイクルアカウンティングの積み上げグラフ

上記の例は一部を切り出した単純な例であるが，今後はより使いやすいデータ出力を目指す．具体的な案として，XML ファイルを解析して CPU 性能情報を表示することをブラウザや Eclipse のプラグインで実施することを検討している．

## 6. おわりに

本稿では，プロファイラとして情報取得を行う精密 PA 機能の汎用化のため，データ形式を見直し XML 化する試みと，オープンソースソフトウェアから取得した情報を加工する仕組みを開発した．

Score-P を利用することで，他のシステムへの利用が可能となる CPU 性能計測の基盤を開発できた．現在は「京」や FX10 および FX100 においてのみ利用可能であった精密 PA 機能を，他のシステムへと転用可能となる道筋を示すこと

ができた．また，Score-P の情報を利用するために，プロファイルデータを XML 表記へと加工する XMLconverter を開発した．XMLconverter の性能を評価したところ，XML ファイルの加工時間の長さやデータサイズの大きさの 2 点で改善の必要性が明確になった．

XML 形式にした性能情報を表示する仕組みについては開発の余地があるため，今後の課題解決に向けて取り組んでいきたい．

## 参考文献

[1] SCORE-P: http://www.vi-hps.org/projects/score-p/
[2] Tuning and Analysis Utilities: http://www.cs.uoregon.edu/research/tau/home.php
[3] MAQAO: http://www.maqao.org/
[4] Vampir: https://www.vampir.eu/
[5] 高瀬亮, 横川三津夫（編）: 情報処理, Vol.53, No.8, chapter 特集: 京速コンピュータ「京（けい）」, 一般社団法人 情報処理学会(2012).
[6] PRIMEHPC FX10: http://www.fujitsu.com/jp/products/computing/servers/supercomputer/primehpc-fx10/index.html
[7] FUJITSU Supercomputer PRIMEHPC FX100: http://www.fujitsu.com/jp/products/computing/servers/supercomputer/primehpc-fx100/
[8] 井田圭一，大野康行，井上俊介，南一生：スーパーコンピュータ「京」の性能プロファイルとデバッグ，FUJITSU VOL.63, NO.3, pp.305-312 (2012).
[9] SPARC64 VIIIfx Extensions 日本語版 （2010 年 4 月 26 日） http://www.fujitsu.com/downloads/JP/archive/imgjp/jhpc/sparc64viiifx-extensionsj.pdf
[10] Pavel Saviankou, Michael Knobloch, A. Visser, Bernd Mohr: Cube v4: From Performance Report Explorer to Performance Analysis Tool. Procedia Computer Science, 51:1343–1352, June 2015.
[11] NAS Parallel Benchmarks: https://www.nas.nasa.gov/publications/npb.html
[12] YAML: http://yaml.org/
[13] Efficient XML Interchange (EXI) Format 1.0 (Second Edition): https://www.w3.org/TR/exi/