社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－HPC－111 (34)
2007／8／3

自動チューニングにおける実験計画的手法を考慮した性能評価データ収集器の設計と実装

小谷和正† 須田礼仁†

本稿では、ソフトウェア自動チューニングにおける実験計画的手法に関する研究の一環として、共通に用いることのできる性能データ収集ソフトウェアシステムの提案を行う。提案するシステムは、ループアンローリングなどのような問題固有の機能を省き、単純な仕組みで拡張性の高いソフトウェアを目指している。具体的には、システムは実験計画モジュールと性能データ収集モジュールに分かれる。前者はユーザが与えたパラメータ定義から試行パターンを生成し、試行により得られたデータから統計的解析などを用いて最適パラメータを決定する。後者は生成された試行パターンとユーザによる設定ファイルのテンプレートに従って対象プログラムの設定ファイルを生成し、それを実行し、そして得た性能データを DB に格納する。また本稿では、Scalapack のサンプルプログラムを題材として実装の試作を行い、提案システムによる自動チューニングの実現を示す。

Design and implementation of Performance Data Collector for Automatic Tuning with Experimental Design Approach

KAZUMASA KOTANI † and REIJI SUDA †

In this paper, as part of studies on experimental design for automatic performance tuning, we propose a common software system of performance data collector. Instead of cutting out domain-specific functions such as loop unrolling, our proposal system is designed to be a simple and scalable software. Specifically, the system consists of two modules, experimental designer and performance collector. The former generates configuration patterns from parameter definitions written by user, and estimates optimal configuration from execution data, with statistical analysis and so on. The latter, according to each of those patterns, generates the configuration file from its template for target program, executes it, and stores its performance data to DB. Moreover, this paper, with taking up the sample program of Scalapack as an example, shows a prototype implementation of the system and its tuning operations.

1. はじめに

従来の自動チューニングソフトウェアでは、パラメータの組合せ爆発によるインストール時間の増大が問題となっており、より低コストでチューニングを行う手法が求められている。

我々の研究[1]では統計学的な手法の一つである実験計画法を性能チューニングに用いることを検討しており、またその他にも田中ら[2]や須田[3]による研究等がなされているものの、万能な手法が存在するものではない。そこで本研究では、今後さまざまな手法 (本稿では上記のような手法をまとめて**実験計画的手法**と呼ぶ) が研究されることを考慮し、共通に用いることの可能な基盤ソフトウェアを構成できれば有用であると考え、そのようなシステムを開発することとした。

自動チューニングを実現する既存のフレームワーク、例えば片桐らによる ABCLibScript[4] などは、実験計画的手法との組合せについてはあまり考慮されていない。ABCLibScript は性能モデルとなる関数を与えることはできるが、用意された機能以上の拡張は容易ではない。自動ベンチマークツールであるが、提案システムと類似のものとして Wright らによる Auto-pilot[5] がある。これはスクリプトによってベンチマークの繰り返し処理を記述できるものであるが、本研究はこれをさらに汎用化し、最適パラメタ推定の議論を含めたものと言ってもよいだろう。

本稿ではまず提案システムのターゲットについて説明する。次に設計と試作実装について解説を行う。例題としては Scalapack のテストプログラム `xdlu` を用いた。

† 東京大学大学院 情報理工学系研究科
Graduate School of Information Science and Technology, University of Tokyo

2. 提案システムの概要

2.1 対　　象

本システムは、プログラム一般に対するチューニングを可能にすることを目標としている。ライブラリ関数やアルゴリズムにおけるパラメタ設定といったものは自明な置き換えであるが、ループアンローリングなどプログラム変換のように複雑なものについては、本システムが提供するチューニング機構と開発者の用意する変換機構の組み合わせによって実現するという形態を想定している。

2.2 システム概要

本システムではチューニング機構全体を二つの部分、つまり**実験計画モジュール**と**性能収集モジュール**に分離している。前者は基本的に実験パターンの生成と最適かルーチンの提供、後者は対象プログラムへのパラメタ入力と性能データ収集や保存を担当する。

本研究は、性能収集モジュールについては共通の実装を与え、実験計画モジュールを開発者 (や研究者) が自由に実装することができるという仕組みを提案することを目的とする。

なお今後の課題として後の節で触れるが、現時点では実行前のチューニングを主なターゲットとしているものの、実行時チューニングについても本質的な原理は同一に実現可能であると考えている。

2.3 チューニング制御

提案システムによるチューニング操作は、「対象プログラムに対する何らかの設定ファイルについて、その値を書き換える」ことで実現する。

より具体的には、ユーザがまず以下を記述する。

(1) パラメタ名と候補値を列挙したパラメタ定義ファイル

(2) 定義したパラメタ名を設定ファイルの置き換えるべき部分に記述したテンプレートファイル

これらを本システムに与えると、実験計画モジュールが実験パターン (パラメタ値の組み合わせ) を生成し、それに基づきデータ収集モジュールは設定ファイルの生成と対象プログラム実行を自動的に繰り返す、という動作をする。

実用上は、対象プログラム実行の前後に行うセットアップ処理とクリンナップ処理の概念を導入すると有用である。例えば設定ファイルとしてプログラムのソースコードそのものを与えるならば、セットアップ処理としてコンパイルを行えばよい。またセットアップ処理にプログラム変換ルーチンを含めるなどすれば、より複雑なチューニングへも拡張し得る。なお、本実装ではそれぞれ単純にコマンドを与えるだけの実装とした。

2.4 データベース

各チューニングパラメタの値と測定性能値の組を読み書きすることができればよいが、本研究においてはデータベースシステムとして主に関係データベース (RDBMS) を用いることを考えている。これは、各モジュールによるデータ抽出を SQL 文によって簡潔に表現することができること、また場合によっては SQL の統計関数を利用できることなどの利点があるためである。

RDBMS の実装は数多く存在し用途や制約にあわせて選択可能である上に、その API については ODBC や JDBC など共通化がなされていることも利点である。例えば MySQL などを使い、データは統一的なサーバ上に集計しつつベンチマークを分散実行するといったことは困難なことではない。

3. 設計と実装

改めて、提案システムの設計や実装における基本方針は以下の二つである。

- 実験計画手法の置き換えなど、拡張を容易にする。全体は小さな機能単位に分離分割されているべきで、各機能間のインターフェイスは簡潔で再利用に適した形態をとるべきである。
- 既存の様々なソフトウェアについてもできるだけ適用可能であるようにする。対象プログラムの言語は限定しない。そのため、プログラム中に埋め込むのではなく外部のツールとして実行可能な設計とする。またソースコードにもなるべく手を加えずに済むことが望ましい。

提案システムの構成図を**図 1** に示す。 以下ではシステムの各部分について試作実装を含め解説を行う。

3.1 性能データベース

提案設計ではデータ型として INTEGER(整数), REAL(浮動小数点数), TEXT(文字列:標準 SQL では CLOB) のみを扱うこととした。

テーブルはパラメタテーブルと評価値テーブルの二つを作成する。つまり、

- パラメタテーブル : 実験番号, パラメタ *1* の値, パラメタ *2* の値,...
- 評価値テーブル : 実験番号, 評価値 *1*, 評価値 *2*,...

として、実験番号によって関連づける。SQL によるデータ操作については後の節で例を示す。

なお本実装においては RDBMS として、特別な準備なくスタンドアロンで利用できる SQLite[6] を用い

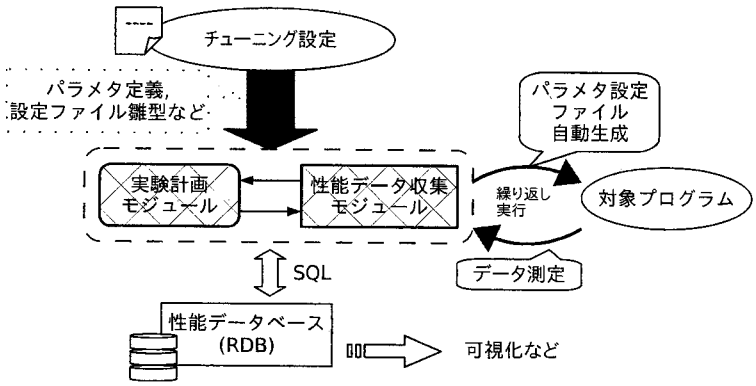

図 1 提案システムの構成

ている。SQLite は近年ではスクリプト言語なども含め多くの言語でサポートされている。

3.2 実験計画モジュール

実験パターンの生成と最適化ルーチンの提供を行う。その他、パラメタ組合せの禁則処理なども可能であろう。

詳細は各々の実験計画的手法により異なるが、外部インターフェイスについては次のようなものである。

パフォーマンス測定時には、入力としてパラメータ定義や手法固有の設定、必要ならば既存の測定データを受け取り、それらを基に行うべき実験パターン (群) を出力する。また最適化時には同様に、最適なパラメタパターンを推定し出力すればよい。

既存データの読み出しについては、データベース (ファイル) 名などを与えられてそれを直接扱うことができればインターフェイスとしては簡潔である。

また、実験パターンついても前項で述べたパラメタテーブルにエントリを追加することを設定パターンの出力と見なすことができる。つまり、データ収集モジュールは、評価値のエントリがない番号のパラメタ設定について実験を新たに行うこととすればよい。ただし、他システムとの組み合わせなども考慮すると、ラッパーの用意や CSV による出力などのサポートは当然必要であろう。

3.3 性能データ収集モジュール

出力ファイルの雛型から実ファイルの生成については、基本的にパラメタ変数名で記述されるプレースホルダ部分の置き換えで実現する。ただし多少柔軟性を持たせるために、「指定したコマンドを実行してその結果の値で置き換える」機能も実装した。通常のプレースホルダは`${NB}` という書式になっているが、例えば `${‘python -c "print $(NP)/$(P)"‘}` などと記述すると、パラメタ NP/パラメタ P の値で置き換える、ということができる。

現実装ではデフォルトで対象プログラムの開始から終了までの時間を計測する。よく使われる計測機能については本モジュールで用意するべきであるが、それ以外にも対象プログラム側で計測や計算した何らかの値を用いることも必要であろう。そこで実装としては、対象プログラムの (パイプ) 出力を計測値として用いる仕組みを用意することとした。また、パラメタ定義ファイルにおいても同様の仕組みを用意した。

対象プログラムの前後にセットアップとクリンナップの処理を行うという考え方は Auto-pilot と同様であるが、そこに自由にコマンドを指定できるるだけで様々なことが実現できる。例えば ABCLibScript や gprof などのように計測機構を含む他のツールと組み合わせる場合には、セットアップ処理などでそのツールによるプログラム実行と計測まで行ってしまい、その出力ファイルを処理し必要な値を抽出するスクリプトなどを計測対象プログラムに指定するということも可能である。

さらに逐次に実験を追加することも想定するため、実験パターンセットの問い合わせは実験計画側で完了を判断するまで繰り返し行う。

実装は C++により行った。補足であるが、対象プログラム実行やデータベースアクセスなど、内部 API はアプリケーションにチューニング機構を直接組み込む際にそのまま用いることができる (言語は限定される)。これは、データ収集モジュールそのものは「別のプログラムを実行する」プログラムであるとみなせることに由来する。

parameters.def

```
% NP = 30          % プロセッサ数
% NB = [40, 80, 120, 160] % BLOCKSIZE
% P = `python -c "for x in [(i) for i
  in xrange(1, $(NP)+1) if $(NP)%i == 0]:
     print x"`  % process grid=NP の約数
```

図 2 パラメタ定義ファイル

LU.dat.tmp

```
'SCALAPACK, LU factorization input file'
...(途中略)...
${NB}             values of NB
...
${P}              values of P
${`python -c "print $(NP)/$(P)"`}
                  values of Q
...
```

図 3 設定ファイル (LU.dat) の雛型

4. チューニングの実例

Scalapack(http://www.netlib.org/scalapack/) のテストプログラムである xdlu を題材として、本システムの実行例を示す。

チューニングパラメタはブロックサイズ NB とプロセスグリッド $(P \times Q)$ の 2 変数とし、システムで計測した実行時間を評価値とした。

入力として**図 2** のパラメタ定義と**図 3** のテンプレートを記述した。実験計画モジュールとしては、直交多項式モデル (主効果のみで 3 次多項式) へのフィッティングを行うものを簡単に実装した。この上で、

```
> opt mpirun -np 30 xdlu
```

などと実行 (opt が本システムのコマンド) すると、実験パターンと相当する設定ファイルが自動的に生成され、その都度 xdlu が自動的に実行される。結果としては**図 4** のようになり、$NB = 160, (P, Q) = (5, 6)$ が推定最適パラメタとなり、その値による設定ファイルが得られた。これをもってチューニング処理の完了となる。

ここでは多項式の係数がモデルパラメタであるが、例えば NB の主効果の推定には $nb = 40, 80, ..., 160$ について ($NB = nb$ のときの測定値の平均) を必要とする。SQL 文としてはテーブルが

```
CREATE TABLE PARAMS (seq INTEGER PRIMARY
 KEY, NP INTEGR, NB INTEGER, P INTEGER);
CREATE TABLE EVALS (seq INTEGER PRIMARY
 KEY, exectime DOUBLE);
```

と作成されており、データ抽出は

```
SELECT avg(EVALS.exectime) FROM PARAMS,
 EVALS WHERE PARAMS.NB=40 and
 PARAMS.seq = EVALS.seq;
```

のような記述になっている。

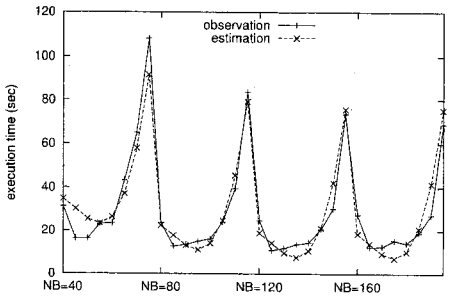

図 4 計測値とモデルによる推定値

5. まとめと課題

実験計画的手法を考慮した自動チューニングシステムを提案し、設計と実装を示した。また、簡単な例によりチューニングの手順を示した。

課題としては、まず本システムで多くの実験計画的手法や実アプリケーションに適用できるかどうかを実証することが挙げられる。またさらなるインストール時間の短縮として、一定の実行時間で試行を打ち切る、といった処理が考えられる。そのように対象プログラムに与える様々な操作が抽象化できるかどうかも検討課題である。

また、ファイル入出力はソケットの入出力へ容易に拡張でき、提案システムをデーモンのようにして動作させることも可能であると我々は考えている。このようなミドルウェア的な実装についても考慮しなければならない。

謝　辞

本研究は、文部科学省科学研究費補助金、特定領域研究「情報爆発」の補助を下で実施しました。また、実験の一部は同提供のプラットフォーム InTrigger をお借りして行いました。

参 考 文 献

1) 小谷和正，須田礼仁：汎用的なソフトウェア自動チューニング機構のための実験計画法の応用の検討，情報処理学会 研究報告, 2006-HPC-107, pp. 193–198 (2006).
2) Tanaka, T., Katagiri, T. and Yuba, T.: d-Spline Based Incremental Parameter Estimation in Automatic Performance Tuning, *Proceedings of PARA'06, CP4* (2006).
3) 須田礼仁：実行時自動チューニングのための逐次実験計画－分散が共通な２つの正規分布の場合－，情報処理学会 研究報告, 2006-HPC-108, pp.13–18 (2006).
4) 片桐孝洋：ソフトウエア自動チューニング–数値計算ソフトウエアへの適用とその可能性，慧文社 (2004).
5) Wright, C. P. et al.: Auto-pilot: A Platform for System Software Benchmarking, *Proceedings of the Annual USENIX Technical Conference, FREENIX Track*, pp.175-187 (2005).
6) Hipp, R.: SQLite home page, `http://www.sqlite.org/`.