ハイパフォーマンス
コンピューティング 72－3
（1998. 8. 6）

自動適応並列プログラム性能最適化ツール TEA Expert

佐藤三久[†] 建部修見[††]
関口智嗣[††] 朴 泰祐[†††]

TEA Expertは、プログラム中に記述した最適化要素に対応したパラメータを変化させ、自動的に実行することによって、最適なパラメータセットを決定し、プログラムの最適化を支援するツール群である。プログラムの最適化の作業を容易にするだけでなく、試行を自動化することによって、異なるプラットフォームにおいても、自動的に最適なパラメータ値を検索することにより、そのプラットフォームに容易に適応できるようになる。ワークステーションからPC、ワークステーションクラスタからMPPなど、異なるプラットフォームごとに逐次および並列プログラムを自動的にチューニングされるソフトウエアのフレームワークを提供する。

Automatic Adaptive Performance Tuning Tool, TEA Expert

MITSUHISA SATO ,[†] OSAMU TATEBE ,[††] SATOSHI SEKIGUCHI [††]
and TAISUKE BOKU[†††]

TEA Expert is a set of tools which supports the performance tuning process for sequential and parallel high performance software by automating exhaustive execution to search the optimal set of tuning parameters. It makes the high performance software automatically adaptive to different platforms as well as supporting the performance tuning process. It provides a framework for designing automatically tuned software adaptive for a wide range of platforms, such as from PCs to workstations, or from workstation clusters to MPPs.

1. はじめに

我々は並列プログラムの性能に影響を及ぼすパラメータを最適化(チューニング)するツール環境TEA Expertを構築中である。本稿では、その基本的なアイデアと設計方針について述べる。

並列、逐次を問わず、プログラムの性能をチューニングする場合、最適化コンパイラを用いることはもちろんであるが、時間のかかる部分を把握し、プログラムを書き換えたり、性能に影響を及ぼすパラメータを変えながら、試行し、最適値を見つける作業を繰り返すことになる。

TEA Expertはプログラムにannotateされた最適化要素に対応したパラメータを変化させて、実行することによって最適なパラメータセットを決定しプログラムの最適化を行うツール群である。例えば、最適化要素に対応するパラメータとは以下のような例が考えられる：

- 行列乗算など、ブロック化できるループのblocking factor。
- 最内周ループのunrollingの回数。
- 並列プログラムでのデータの分割方法とサイズ。
- ループの並列実行時のスケジューリングの方法。スケジューリングを動的に行うか、静的におこなうか、など。

このようなパラメータを性能パラメータに呼ぶことにする。逐次プログラムだけでなく、特に並列プログラムにおいては、そのプラットフォームに依存する様々な性能パラメータがある。TEA Expertはプログラマによって指定された性能パラメータを自動的に適用することによって、最適なセットを決定するシステムである。

最近、新しい高性能マイクロプロセッサが次々と登場している。同じ命令セットを持つプロセッサでも、実行方式、命令のサイクル等でいろいろなバージョンがある。また、同じバージョンのプロセッサを使っていても、キャッシュサイズが違っていたり、チップセットが異なっているなどの違いがある。このような現状において、あるプロセッサにおいて、最適なプログラムでも、他のプロセッサにおいて、そのままのコードで最適な性能が得られるとは限らない。さらに、プロセッサの移り変わりが激しいため、最新のプロセッサの情報を得るこ

† 新情報処理開発機構
Real World Computing Partnership
†† 電子技術総合研究所
Electrotechnical Laboratory
††† 筑波大学
University of Tsukuba

とが難しくなってきており、コンパイラの最適化も対応していないことが多い。例えば、初期の pentium のコンパイラは、i386 向けであり、命令のスケジューリングなど十分対応されていなかった。

また、コンパイラによっては決定できないパラメータも多い。例えば、キャッシュのサイズに依存するものについてはあらかじめコンパイラに組み込んでおくこと困難であり、データのアクセスパターンを把握する必要があるため、プログラマによる指示あるいはヒントが必要である。現在の多くのコンパイラはプログラムの構文解析などの静的な情報しかつかっていなく、キャッシュサイズなど、実行してみなくてはわからない動的な情報についてはユーザが与えてやらなくてはならない。

プログラムのソースコードが他のプラットフォームでコンパイル実行できることを移植性と呼ぶが、さらに、性能を低下させずに移行できることを**性能移植性**と呼ぶことにする。TEA Expert は、プログラムの最適化の作業を容易にするだけでなく、試行を自動化することによって、未知のプラットフォームにおいても、自動的に最適なパラメータ値を検索することにより、そのプラットフォームに容易に適応できるようにし、性能移植性を高めることができる。

現在、特に、このような技術が必要とされているのは、BLAS などの数値計算の基本カーネルになるルーチン群である。ATLAS[6] は、行列乗算ルーチン (dgemm) に限って、ブロッキングのサイズなどを変えて自動的に実行することによって、そのプラットフォームに対応するルーチンを生成するプログラムである。これによって、ベンダ提供のルーチンに匹敵する性能を持つルーチンが生成できることが報告されている。TEA Expert は、ここでの手法を特定のルーチンに限らず、一般的なプログラムに適用できるように拡張しようというものである。

主に数値計算ルーチンを対象に TEA Expert のプロトタイプを試作した。次章において、これまでの性能解析ツールとチューニングソフトウエアについて、概観する。3 章において、TEA Expert の概要について述べ、4 章で試作した TEA Expert システムと適用例について述べる。5 章でこれからの課題について述べる。

2. 性能解析ツールと自動適応チューニングソフトウエア

これまで、多くの性能解析ツールが開発されてきた。ツールを使うことによって、並列プログラムの挙動について、解析が可能である。プログラムのチューニングする場面において、使用されるツールについて、以下のようなものがある：

- 並列プログラムの実行時の通信、プロセッサの状況、プログラマが指定する様々なイベントについてデータを収集、可視化、ユーザの実行状況の把握を補助するツール。通信状況を表示するツールには MPI の upshot、pvm の xpvm、vampire などがある。Pablo[4] では、ユーザが指定したイベントに関して柔軟な可視化をすることができる。
- 性能可視に加えて、性能診断のための知識を蓄えておき、性能データから診断を行い、ユーザを補助するツール。代表的なツールとして Paradyn[3] がある。
- コンパイラが、静的にプログラムを解析することによって、性能を推定し、並列プログラムの構成を最適化を自動、もしくは半自動に行うシステム。P^3T[2] や PCASE[5] がある。
- 特定のプログラムについて、自動的に試行することよって、最適化されたソフトウエアを生成するシステム。ATLAS[6],PHiPAC[1] など、数値計算にとって、最も重要なルーチンの一つである行列乗算 dgemm を各プラットフォームに最適なパラメータ (blocking/tiling factor+ latency) を見つけ、生成してくれるプログラムである。

性能解析ツールの問題点は性能の解析を容易にしてくれるものの、チューニングは人手で行わなくてならないところである。解析を容易にするツールとして、性能診断ツールがあるが、現実の問題は複雑であり、実用化に成功しているとは言いがたい。これらツール全般に操作が習熟するのに時間がかかり、実用上の問題となっている。

性能解析結果を把握するのももちろん重要であるが、この結果を自動的に反映してチューニングするシステムが期待されている。コンパイラと統合し解析するシステムが最も期待される手法であるが、コンパイラが読み切れない場合には何らかの実行状況に関する情報が必要であり、プログラマからの情報が必要になっている。

3. TEA Expert の概要

TEA Expert の設計目標を以下に挙げる：

- 性能解析評価によって得られる情報をなるべく人手をかけずにプログラムの最適化に反映できる環境を目指す。
- プログラムをチューニングする場合には、プログラマは性能パラメータなどを変えながら、試行実験を行わなくてはならない。そのチューニングを支援するような実験環境を提供する。
- 結果として、実行時間が最小のものを選ぶことによって、自動的に実験対象となっているプラットフォームに対して、最適なパラメータセットを持つプログラム (ルーチン) が生成できる。
- ATLAS や PHiPAC のように、行列の乗算に特化したものでなく、コンパイラと統合することによって、より一般的なプログラムに適用できるような枠組とする。

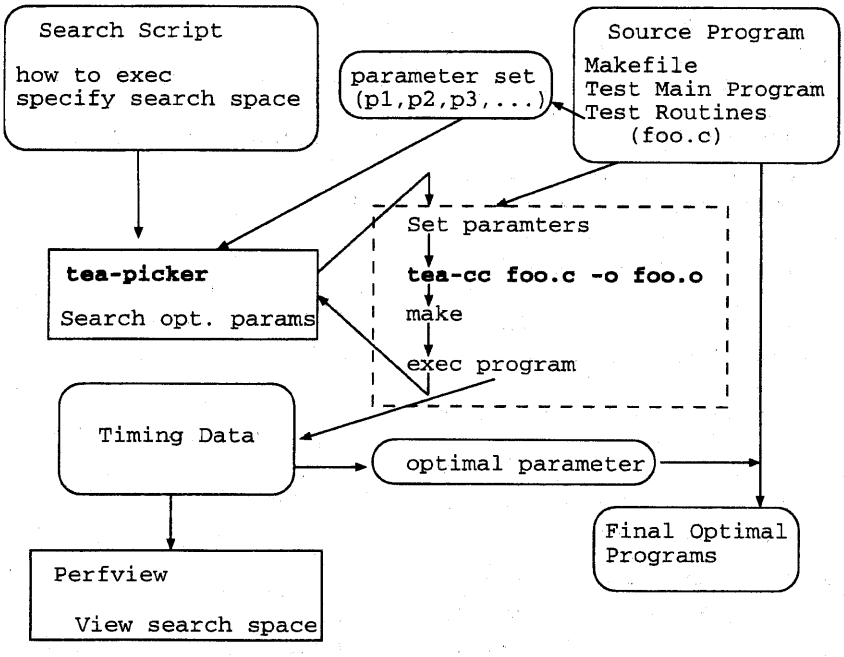

図1 TEA Expert の構成

- 通信速度やキャッシュの容量など、マシンの特徴量を定量的に計測できる環境としても利用し、これをプログラム最適化に用いることのできるデータベースにする。パラメータを変化させて、実行、測定を行う作業をサポートする。

まずは、手始めとして、C プログラムを対象とする自動適応並列プログラム性能最適化ツール TEA Expert を試作した。基本的な設計方針を以下のようにした：

- 現在のところ、数値計算ライブラリなどの最適化を対象とする。
- 与えられた問題サイズ、入力データについて、最適なプログラムを生成することを目指す。
- 性能の測定は静的な環境で行われることを前提とする。動的に変動する環境での最適化は行わない。

図1に構成を示す。

- source program:
  - プログラムをコンパイル、リンクするための Makefile
  - テストのための main プログラム。このなかに、データのセットアップ、pragma で指定された計測区間の定義が入る。
  - チューニングするべきルーチンのソースファイル (foo.c)
  - その他のプログラム
- parameter set:
  チューニングするべきパラメータの集合。ソースの中 (foo.c) に pragma で annotate されており、これをコンパイル時に検出する。
- search script:
  - どうやってプログラムを実行するかの指定。
  - パラメータに関しての検索空間を記述する。
- tea picker:
  パラメータをセットし、コンパイル、実行をするプログラム。
- timing data:
  "(実行時間、パラメタの値、....)" という形の実行結果のレコードを持つデータベースをつくる。
- perfview:
  timing data を可視化して、様子をみるグラフツール。
- optimal paramters and final program:
  最終的に timing data から、最良の値をえらび、プログラムをコンパイルし、最終的なバージョンを作る。

大体の手順は以下のようになる：

(1) 最適化する対象プログラムを用意する。性能パラメータについては、**#pragma TEA** から始まる directive をつかって、性能パラメータの情報について、プログラム中に annotate する。

(2) 性能の入力データや測定区間をセットアップするテスト用のメインプログラムを作成する。

(3) search script を作成し、検索するパラメータ空間を指定する

(4) tea-picker により、測定区間を最小にする最適な

パラメータを決定する。

(5) 最適なパラメータを用いて、再度、コンパイルし、それをインストールする。

4. TEA Expert の詳細

ここでは、例を示しながら、各部について説明する。

4.1 ソースプログラムへの annotation とコンパイラフロントエンド tea-cc

tea-cc は、性能パラメータに関して annotate された C プログラムを入力として、性能パラメータを設定したプログラムを生成するコンパイラフロントエンドである。**#pragma TEA** から始まる directive を解釈し、C コンパイラを起動し、コンパイルを行う。

現在サポートしている directive には以下のものがある：

- パラメータの指定。*param1,param2,...* の識別子が性能パラメータであること宣言する。tea-picker で指定された値に置き換えられる。

  #pramga TEA paramter(*param1, param2, ...***)**

- 測定区間の指定。ここで指定された区間の実行時間が最小になるように、パラメータを探索する。この箇所に性能測定のためのタイマールーチンと結果の出力ルーチンを埋め込む。

  #pramga TEA begin
  #pramga TEA end

- 条件コンパイル。パラメータに関して、条件が成立した場合に有効になる。

  #pramga TEA if(*condition***)**
  #pramga TEA endif

- ループの unroll の指定。loop unrolling は数値計算ルーチンにおいて、性能に影響を及ぼす重要プログラムの最適化技法であるため、tea-cc で変形をサポートする。但し、直後に unroll が可能な for ループがなくてはならない。unroll のパラメータが複数の場合はループ交換が可能であるとみなし、多重のループのブロッキングを行う。また、**letency** は、unroll された文間のスケジューリングのためのパラメータである。

  #pramga TEA unroll(*p1, p2, ...***)** latency(*p*)

図 2 に行列乗算ルーチンの例について示す。性能パラメータは行列 A をキャッシュブロッキングするサイズ**BLK_N** と最内周ループの unroll 回数**N_UNROLL** である。なお、このルーチンは文献[8] 中のルーチンを参考に作成した。

4.2 最適パラメータサーチプログラム tea-picker

tea-picker は、search-script で与えられた性能パラメータを変化させ、プログラムを実行させる。

```
/* test main routine */
#define N 500
double A[N*N],B[N*N],C[N*N];

main()
{
   ... set up ...
#pramga TEA begin
   dMM(A,N,B,N,C,N,N,N,N);
#pramga TEA end
}

#pramga TEA paramter(BLK_N,N_UNROLL)

dMM(double a[],int lda,double b[],ldb,
  double c[],int ldc,int l, int m,int n)
{
  int i,j,k,ind;
  int kk,ll,ii,mm,ms,ls;
  static double work[BLK_N*BLK_N];
  ll = min(l,BLK_N);  mm = min(m,BLK_N);
  /* K block size = mm */
  for(kk = 0; kk < m; kk += mm){
    ms = min(m-kk,mm);
    for(ii = 0; ii < l; ii += ll){
       ls = min(l-ii,ll);
       ... pack to work ...
dMM1(work,&b[kk],ldb,&c[ii],ldc,ls,ms,n);
    }
  }
}

dMM1(double a[],double b[],int ldb,
  double c[],int ldc,int l,int m,int n)
{
  int i,j,k; double t;
  for(j=0; j<n; j++){
#pragma TEA unroll(N_UNROLL)
    for(i=0; i<l; i++){
       t = c[j*ldc+i];
       for(k=0; k<m; k++)
          t += a[k*l+i]*b[j*ldb+k];
          c[j*ldc+i] = t;
      }
    }
}
```

図 2 TEA Expert プログラム例 (行列乗算)

search script には、以下のものが記述できる。

- **Execute:** *program-name arg1 arg2 ...*
  テストプログラムの実行方法を指定する。指定されない場合は、**a.out** を実行する。
- **Make:** *make commands*
  テストプログラムを make するためのコマンドを指定する。指定されない場合には、**make** コマンドを実行する。
- **Search:** *parameter range*
  性能パラメータの探索空間を指定する。*parameter range* は、以下のもので、これらの積&、和| を指定できる。
  - 範囲指定 param=[min,max,step]
  - 列挙指定 param={value1,valu2, ...}
  - 値指定 param = value

  パラメータの指定については、ある特定のパラメータの場合に探索の仕方を変えるなどの複雑なケースが考えられるが、現在は単純なケースのみサポートしている。

tea-picker は、まず、tea-cc を用いて、どのファイルにどのパラメータがあるかを調べ、パラメータと実行ファイルの対応のデータベースを生成する。パラメータを変化させたときに、どのオブジェクトを無効にするべきかを決定する。無効にされたオブジェクトは Make コマンドによって、再度コンパイルされる。

script に指定された探索空間で性能パラメータの組合せを変えて、プログラムを実行する。実行されたプログラムの結果は、パラメータの値と実行時間の組として、実行結果データベースに記録される。

図 2のプログラムについて、Pentium Pro プロセッサ (200MHz) において、サイズ 500 × 500 の行列を入力データにして、**BLK_N** を 10 から 100 まで 10 刻みで、**N_UNROLL** を 1 から 9 まで 1 刻みで変化させ、実行した。その結果、**BLK_N** が 30、**N_UNROLL** が 6 において、97.2412 MFlops が最高であった。

4.3 性能パラメータ可視化ツール perfview

perfview は、gnuplot を利用した実行結果を可視化するためのツールである。このツールでは、実行結果データベース内のデータに対しパラメータを変化させた場合の実行時間をグラフとして表示する。2 つのパラメータを変化させて、3 次元の表示をすることもできる。図 3にその GUI を示す。

図 4 に、図 2 のプログラムの前節の実行結果について 2 つパラメータに関する実行時間の変化を可視化した場合の例について示す。これは、gnuplot からの tif 出力である。X window 上では、3 次元空間で回転などをしながら、結果を見ることができる。

TEA Expert は、自動的に全てをバッチ的に使用することも可能であるが、探索空間を適宜設定することによって、一つの性能チューニングの実験環境としてとして用いることができる。実行時間だけでなく、FLOPS

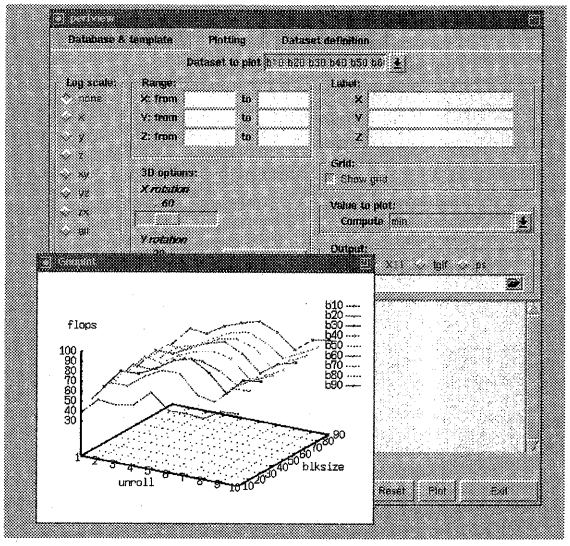

図 3 perfview の GUI

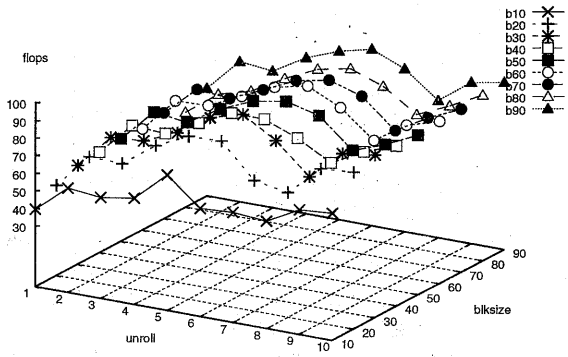

図 4 perfview の 3 次元表示の例

値や speedup/effciency など、性能評価に関する加工をサポートしている。また、探索空間を効率的に設定するためにもパラメータと実行時間を可視化して見ることは有効である。

4.4 高精度の性能測定用タイマールーチン

最近の高性能マイクロプロセッサには、クロックレベルの性能測定用のカウンタをもっているものが多い。精密なカウンタを使うことによって、短い時間での正確な時間計測が可能になり、外乱の影響を極力さけることができる。

現在、TEA Library[7] の一部として、各マイクロプロセッサ対応の共通の測定ルーチンを実装中である。なお、各主要プロセッサのクロックカウンタ関連のハードウエアの情報は、

http://www.rwcp.or.jp/lab/pdperf/timer-collection

にまとめてある。

4.5 最適ルーチンの作成・インストール

最終的に探索空間のなかで、実行時間が最小のパラメータを組合せを選択して、再度コンパイルすることによって、最適なルーチンを得ることができる。tea-picker に -final オプションをつけて実行することによって、tea-cc によって設定されるパラメータは実行結果データベース中で実行時間が最小のものになる。この状態で、通常のコンパイル、インストール手続きを実行すればよい。

5. 今後の検討課題

TEA Expert は、性能に影響を及ぼすパラメータをチューニングする作業を自動化することによって、実行されるプラットフォームに対し、自動的に最適するソフトウエアを目指すものである。

試作した TEA Expert ではライブラリとして用いられるプログラムに対して、実行時の変動が少ない静的な環境下において、特定の入力データに対してプログラム中で annotate された最適化パラメータを決定することを目的としている。

これからの検討課題として以下のものがある：

- 大規模なプログラムでは、単純な exhaustive なパラメータ検索では膨大な時間がかかってしまう。そのために、検索するべきパラメータ空間を何らかの戦略で効率的に検索する必要がある。この戦略について、記述する方法、あるいは自動的に行う戦略を検討しなくてはならない。
- 試作したシステムでは性能変動が少ない静的な環境を仮定しているが、そのような環境だけではない。なるべく、動的な環境についてもロバストになるように対処しなくてはならない。
- 問題サイズなど入力パラメータに対応したコードのバージョンの自動生成。試作したシステムは、設定された入力データに対して最適化を行うが、複数の入力データに対して対応するように、コードを複数用意する変形を自動的に行う方法などが考えられる。
- コンパイラで読み切れない、実際に実行してみなくては分からない情報を最適化するための環境。コンパイラから、試行してみるパラメータを生成できれば、そのパラメータはこのツールをつかって、決定できる。
- P^3T や PCASE で行われているようなコンパイラによる静的な解析による性能予測技術との統合。これにより、性能パラメータの検索空間を絞ったり、上に述べたコンパイラからの性能パラメータの抽出ができる可能性がある。

謝　辞

日頃、性能評価技術に関し議論して頂く TEA グループのメンバー諸氏に感謝いたします。TEA グループは、研究技術組合新情報処理開発機構・電子技術総合研究所・筑波大学を中心とする性能評価に関する研究グループである。

参 考 文 献

1) J. Bilmes, K. Asanovic, C.W. Chin, and J. Demmel. Optimizing Matrix Multiply using PHiPAC: a Portable, High Performance, ANSI C Coding Methodology. In *Proceedings of the International Conference on Supercomputing*, pages 340–347, July 1997.
2) T. Fahringer. Estimating and Optimizing Performance for Parallel Programs. *IEEE Computer*, 28(11):37–56, November 1995.
3) B.P. Miller, M.D. Callaghan, J.M. Gargille, J.K. Hollingsworth, R.B Irvin, K.L. Karavanic, K. Kunchithapadam, and T. Newhall. The Paradyn Parallel Performance Measurement Tool. *IEEE Computer*, 28(11):37–46, November 1995.
4) D.A. Reed, K.A. Shields, W.H. Scullin, L.F. Tavera, and C.L. Elford. Virtual Reality and Parallel Systems Performance Analysis. *IEEE Computer*, 28(11):57–67, November 1995.
5) Y. Seo, T. Komatchi, K. Kusano, Y. Watanabe, and Y. Shiroto. PCASE: A Programming Environment for Parallel Supercomputers. *Parallel Language and Compiler Research in Japan (L,B Bic, A. Nicolau and M. Sato eds.)*, pages 377–404, 1995.
6) R.C. Whaley and J.J. Dongarra. Automatically Tuned Linear Algebra Software. Technical report, University of Tennessee, 1997. http://www.netlib.org/utk/project/atlas.
7) 関口、佐藤、中田、長嶋、朴、中村. 並列分散システム性能評価用ツール群 – tea library – の設計. In 情報処理学会 *HPC* 研究会報告 *60-15*, pages 83–88, March 1996.
8) 寒川 光. *RISC* 超高速化プログラミング技法. 共立出版, 1995.