超低消費電力高性能計算に向けた取り組み

大島 聡史[1,a)]　Luo Cheng[2]　平澤 将一[3]　片桐 孝洋[1]　須田 礼二[2]　本多 弘樹[4]

概要：次世代のスーパーコンピュータに向けて，また高性能なパーソナルコンピュータやワークステーションの実現において，消費電力を下げつつ高い性能を得る超低消費電力高性能計算（Ultra Low Power High Performance Computing : ULPHPC）技術が重要となっている．さらに昨今では国内電力事情を考慮したピーク電力の削減や消費エネルギーの削減など，省電力な計算機を構築し運用する技術への要求が高まっている．我々のグループでは CREST プロジェクト「ULP-HPC: 次世代テクノロジのモデル化・最適化による超低消費電力ハイパフォーマンスコンピューティング」（研究代表者：松岡聡 東工大教授）において，超低消費電力ハイパフォーマンスコンピューティングの実現に向けた電力測定方法の開発，電力測定 API の作成，電力情報を用いた自動チューニング技術の開発とアプリケーション適用といった研究を行ってきた．本稿では ULPHPC に向けた我々の研究グループにおける取り組みおよびその成果について述べる．

Research activity for Ultra Low Performance HPC

Satoshi OHSHIMA[1,a)]　Luo Cheng[2]　Shoichi HIRASAWA[3]　Takahiro KATAGIRI[1]　Reiji SUDA[2]
Hiroki HONDA[4]

Abstract: Technologies of ultra low power high performance computing (ULPHPC) which aims obtaining high computing performance with reducing power consumption is getting more and more important for attainment of next generation supercomputers and high performance personal computers and workstations. Moreover, because social requirement of reducing the peak power consumption and consumption energy is strong, the request for technologies of implementation and operation of low power computers is required. We have researched for attainment of ULPHPC in the "ULP-HPC : Ultra Low-Power, High Performance Computing via Modeling and Optimization of Next Generation HPC Technologies" project. In this project, we researched about measurement method of power consumption, development of power measurement API, and development and application of auto-tuning technologies that use power information. In this report, we describe the activities and results of our research of ULPHPC.

1. はじめに

計算機システムの設計と開発において電力に関する課題が重要性を増している．例えばスーパーコンピュータの構築や導入においては，発電機や電源設備など電力に関する制限が大きな制限の一つとなっている．また近年では環境問題や国内電力事情の影響を受けて総電力消費量やピーク電力の削減が求められており，計算機システムの大小に関わらず低消費電力技術や低消費エネルギー技術の重要性が高まっている．

計算機システムの省電力化については国内・海外を問わず様々な分野で数多くの研究や商品開発が行われている．省電力化には様々なアプローチがあり，例えばハードウェアに関しては，低い電圧で稼働するデバイスを開発する，消費電力を保ちながらより高速に動作できるようにして実行時間を短縮し総消費エネルギーを小さくする，待機電力を下げることで総消費エネルギーを小さくする，特定の処理に対して電力効率の良いアクセラレータを開発する，な

[1] 東京大学 情報基盤センター
Information Technology Center, The University of Tokyo
[2] 東京大学 大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[3] 東北大学 大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University
[4] 電気通信大学 大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
[a)] ohshima@cc.u-tokyo.ac.jp

どの例があげられる．またソフトウェアやアルゴリズムに関するアプローチとしては，並列度やメモリアクセスの回数・粒度を変えることでハードウェアの動作を促進または抑制することで低電力化や低消費エネルギー化が行えることがある．また対象の処理に最適なハードウェアを選択する機構を作成するといった例もあげられる．

CREST 領域「情報システムの超低消費電力化を目指した技術革新と統合化技術」[1] においても省電力化について多くの研究者が様々な研究を行ってきた．本稿の著者らも「ULP-HPC: 次世代テクノロジのモデル化・最適化による超低消費電力ハイパフォーマンスコンピューティング」（研究代表者：松岡聡 東工大教授）に参加し，主にアルゴリズムやプログラミング環境からのアプローチを行ってきた．本稿では超低消費電力ハイパフォーマンスコンピューティング (ULPHPC) に向けた我々の研究グループにおける取り組みおよびその成果について述べる．

2. ハイパフォーマンスコンピューティングと省電力化

計算機シミュレーションは実験，理論と並ぶ研究開発の第 3 の科学 (第 3 の手法) として欠かせない存在となっており，より高速な，より大規模な，より正確なシミュレーションが求められている．そのためにはより高性能なハードウェアやソフトウェアが必要である．

ハードウェアの性能を向上させるうえで，電力は強い制限事項となる．例えばスーパーコンピュータのように大規模な計算機システムを構築・運用するには，規模に見合った電力を安定供給できることが必須である．2012 年に正式稼働を開始した理化学研究所 計算科学研究機構のスーパーコンピュータ「京」[2] は，全系システムを稼働させるために 20MW 程度の電力を必要とする．また 2018 年頃の実現を目指して世界中で研究が進められている ExaFLOPS 級のスーパーコンピュータ（1Exa=1000peta,「京」の性能は約 10petaFLOPS である）については，実現における最大の壁の一つとして電力の問題があげられている．既存の技術をスケールして ExaFLOPS を達成しようとすると非現実的なレベルの電力が必要となってしまうため，技術革新が必要とされている．もちろんパーソナルコンピュータやワークステーションについても，さらなる高性能化の要求は大きい一方で消費電力を大きく上げることはできず，より高い電力効率・省電力性が求められている．

計算機システムの省電力性はソフトウェアによっても高めることができる．現在ではノートパソコンなどのバッテリーで稼働する機器を中心に，負荷が小さい際には CPU やメモリなどのハードウェアを低消費電力な待機状態にさせたり，部分的に電力の供給を止めたりする技術が用いられている．こうした技術を最大限に利用するためにはプログラムによるサポートが必要である．例えばマルチコア CPU の活用においては，搭載されているコアを全て使って処理を行ってもスケーラブルに性能向上が得られない場合には，使用するコアを減らして消費電力を抑えることで総消費エネルギーを削減し電力効率を向上させられることがある．計算時のデータの移動をなるべくキャッシュ内に収まるようにしてメモリやストレージへのアクセスを抑えることも消費電力の削減に効果がある．より単純でわかりやすい例としては，消費電力がほぼ一定であると仮定するならば，プログラムの最適化を進めて実行時間を短くすることは総消費エネルギーを削減する効果がある．このように，省電力化・省エネルギー化を行うためにはハードウェアとソフトウェアの協調が重要である．

汎用的な CPU に加えて特定の処理を高効率で実行可能な専用ハードウェアを用いるという手段も多く用いられている．特に HPC の分野においては GPU の活用 (GPGPU[3], GPU コンピューティング) への注目度が高く，すでに様々なアプリケーションの GPU 化が行われている．GPU は CPU と比較して大量の計算コアを搭載しており，メモリ性能も一般的な PC と比べて高い．そのため特に並列性が高い計算問題に対しては CPU よりも非常に高い性能を得ることができる．一方で GPU の備える計算コアは CPU と比べるとシンプルで汎用性に劣るため，十分な並列度を持たないプログラムなどは高速に実行することができない．このように，GPU はどのようなアプリケーションでも CPU と比べて高速・高電力効率に実行できるわけではなく，適切な使い分けが必要である．

現在 HPC の分野で使われている GPU としては，NVIDIA 社の GPU，主に GeForce シリーズや Tesla シリーズが主流である．これらの GPU においてはプログラム作成のために C/C++を拡張した CUDA[4] が多く利用されている．CUDA は言語仕様そのものの独自性は低い一方で最適化を行うには高度な知識も必要であるため，プログラム最適化に必要なコスト (習得や利用の手間) も無視することはできない．そのため，指示文を用いて容易に GPU プログラミングを行うことができる OpenACC[5] への注目も高まっている．GPU の活用は我々のプロジェクトにおいても極めて重要なテーマである．

3. 電力測定方法と API の開発

現実のハードウェアを対象として省電力化を行うには，消費電力を測定しプログラムから参照する仕組みが不可欠である．しかしながら現在使われている PC などの一般的な計算機システムにおいてはバッテリの消費情報や温度情報の取得は可能であることが多い一方，消費電力を取得する仕組みは備わっていない．そこで本稿の著者である Luo・須田らは，GPU を搭載した PC を対象として電力を測定し参照するための仕組みについて研究を行ってきた．詳細な内容は参考文献 [6], [7] にまとめられているため，本

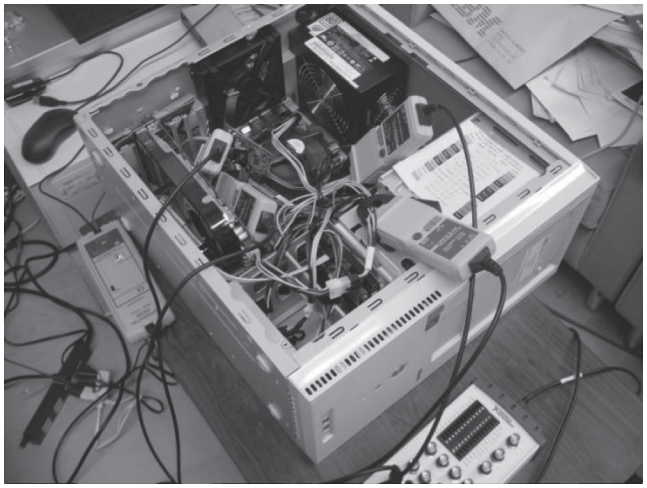

図 1 電源測定用ハードウェアの例

稿では概要のみを述べる．

計算機システムを含む一般的な電気機器の全体的な消費電力については，いわゆるワットチェッカーと呼ばれる機器を用いれば測定することができる．PC 全体の消費電力についても同様に測定可能である一方，この方法では CPU や GPU といったより小さな単位での測定を行うことはできない．PC 内の各パーツへの電力供給については，電源ユニットから各パーツに対して直流電源ケーブルが接続されているため，これらを用いることでパーツ単位での消費電力を測定することができる．直流電源ケーブル単位の消費電力を測定するにはクランプ型の電力測定器 (クランプメーター) が利用可能であり，USB 接続で PC から値を取得できる製品も存在する．

ところで，GPU の提供形態として一般的である PCI-Express カード型 GPU への電源供給については 2 系統で行われている．1 つは電源ユニットから GPU カード上のコネクタに対して直流電源ケーブルを接続する経路であり，もう 1 つは GPU とマザーボードを接続する PCI-Express バスによる電源供給経路である．前者の経路に関する電力についてはクランプメーターにより測定することが可能である．一方で後者の経路についてはクランプメーターを挟むことができないため，他の方法で測定する必要がある．Luo・須田らはライザーカードにより PCI-Express バスの配線を延長し，延長部分の直流電源ケーブルを加工してクランプメーターに挟むことで消費電力を測定できるようにした (図 1)．

さらに Luo・須田らは電力測定による消費電力の変動を防ぐために測定対象計算機と測定を行う計算機を別途用いる設計を採用し，消費電力の測定開始と終了や測定情報をデータベース化して扱う処理などを API としてまとめた (図 2)．これらを用いることで，アプリケーションプログラマは任意のプログラムに対してその処理の一部でどれだけの電力を消費しているのかを測定し，測定結果を利用することが可能となった．

ユーザ用 API の例

```
//marker kernel;
int P_marker();
//send start monitor request;
int P_startMonitor();
//send stop monitor request;
int P_stopMonitor();
//wait for data
int P_waitData();
//receive data
int P_receveData(int sock_fd, struct record* rec);
//store data into database
int P_storeData(struct record* rec);
//half-half benchmark
int P_half_test(int time);
//computing intensive benchmark
int P_compute_test(int time);
//memory access intensive benchmark
int P_memory_test(int time);
//return error imformation
int P_getError();
```

データベースアクセス用 API の例

```
//open database connection
int dbOpen(char* dbName);
//read records by task ID
int dbReadByID(int ID);
//read records by task name
int dbReadByTaskName(char* name);
//qury all the user name in database
int dbGetUser();
//qury all tasks information of user
int dbGetTaskByUser(char* user);
//delete records by task ID
int dbDeleteByID(int ID);
//delete records by task name
int dbDeleteByTaskName(char* taskName);
//delete records by user name
int dbDeleteByUser(char* user);
//close database connection
int dbClose();
```

図 2 開発した API

4. 指示文記述による電力情報を考慮した自動チューニング機構の開発

従来，ベクトル計算機が主流であった際には自動並列化による最適化が主要な技術として用いられた．一方近年では階層性のあるキャッシュを備えたマルチコア CPU が主流となっており，さらに GPU のようなアクセラレータが多く用いられるようになってきたため，自動並列化では良い性能が得られない実行環境 (ハードウェア) やアプリケーションが増加している．しかし任意の実行環境とアプリケーションに対して最大性能を得るには高い技術と手間が必要であり，プログラムの移植性や可読性の観点からも，生産性の高い開発手法やツールへの需要が高まっている．

そこで，指示文を用いたプログラム最適化への注目が高

リスト 1　指示文を用いたプログラム記述の例
(OpenMP によるループ並列化の例)

```
// この例では指示文(pragma omp parallel for)直後の
// forループがスレッドにより並列実行される
int main(){
#pragma omp parallel for
  for(i=0; i<N; i++){
    C[i] = A[i] + B[i];
  }
  return 0;
}
```

まっている．指示文を用いたプログラム最適化においては，プログラムに対してコメントの一種の形式で指示文を挿入し，対応する処理系 (コンパイラやトランスレータなど) を用いてプログラム最適化を行う (リスト 1)．一般的に指示文はプログラムに対してコメントの一種として挿入されるため，指示文を無視すれば非対応のコンパイラでもコンパイル可能であることが多い．またプログラムの構造自体を大きく変更しなくてもプログラムの最適化が行えることや，既存のプログラムからの段階的な適用が行いやすいことも指示文を用いたプログラム最適化が注目される理由としてあげられる．

指示文を用いたプログラム最適化手法としては OpenMP が広く用いられている．OpenMP は共有メモリ型並列計算機を対象としており，主にループ処理の並列高速化に用いられている．GPU に対応した指示文を用いたプログラム最適化手法としては，特に OpenACC への注目が高まっている．また OpenMP を GPU に対応させる研究についても，本稿の著者らによるもの (OMPCUDA[8]) や OpenMPC[9] などがあげられる．

本稿の著者である片桐らは指示文を用いた自動チューニング記述についての研究を行ってきた．片桐らの開発した ABCLibScript[10] は，対象プログラムに指示文を記述することで最適なアルゴリズムの選択などの自動チューニング機能を付加することができる．本研究では片桐らによる既存の研究をもとにして，GPU への対応や電力情報を考慮した自動チューニング機構の開発を行っている．GPU への対応については，図 3 に示すように CPU 向けのコードから電力情報を用いた CPU+GPU 最適化の実現を目指している．現在は手動で記述した CUDA プログラムを用いた最適化や OMPCUDA を用いた GPU 化について検討および実装を行っているが，さらに GPU の性能を引き出すために OpenACC の活用についても検討している．本機構は対応する指示文を記述したプログラムに対して 3 章で述べた電力関連の API を含むソースコードを出力する機能を備えている．そのためアプリケーションプログラマは電力測定に関する知識どころか，電力測定に関する記述自体を行わず

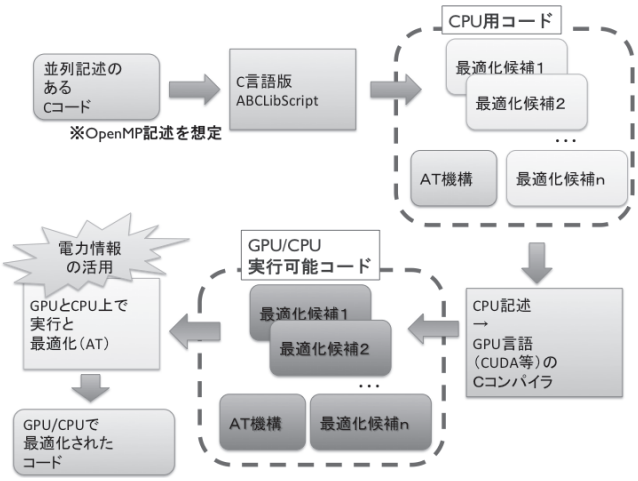

図 3　ABCLibScript による処理の流れ

表 1　実験環境

CPU	Xeon E5-2620 x2 (SandyBridge-E)
メインメモリ	DDR3-1333 4GBx8
GPU	TeslaK10 x4 (8GPUs)
ソフトウェア	CentOS 6.2 x86_64, CUDA 5.0 RC

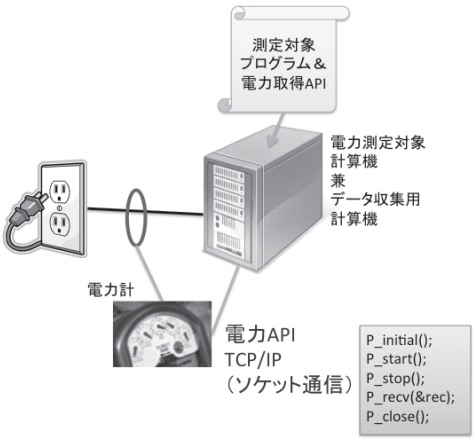

図 4　実験環境の構成

に電力効率の良い実装を採用することが可能となることが期待できる．

5. 評価実験

本章では 3 章と 4 章にて述べた各種の技術を用いてプログラムの分析と最適化を行った例を示す．実験内容としては，反復計算を行うステンシル計算プログラムの主要な計算部分を CPU と GPU それぞれで実装し，開発中の電力情報に対応した ABCLibScript の指示文を用いて消費エネルギーが最少となるようなプログラム実行ができるかを確認した．実験に使用した計算機環境は表 1 の通りである．今回はコンセントと実験機の間にクランプメーターを設置し，PC 全体の電力を測定して消費エネルギー最少化を目指した (図 4)．

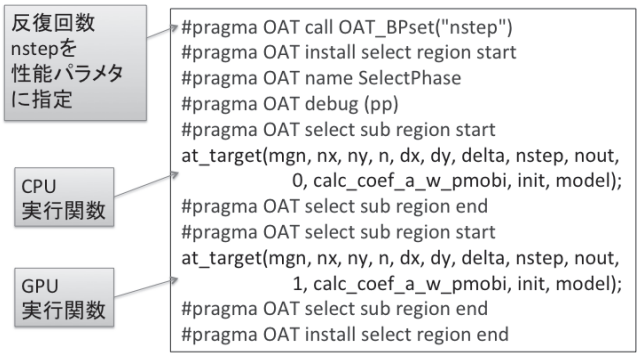

図 5　AT 記述

実験プログラムについて，指示文による AT 記述を図 5 に，自動生成されたコードの主要な部分をリスト 2 に示す．なお生成されたコードには上述の AT 記述に加えて電力測定安定化に用いている ATMathCoreLib[11] の記述も含まれている．このプログラムにより，CPU を用いた場合と GPU を用いた場合それぞれの実行時間及び消費電力が測定される．本実験では反復計算の反復回数を変更した場合に CPU と GPU それぞれの消費エネルギーはどちらが小さいかを比較した．

実験の結果を以下に示す．まず反復計算に対する実行時間と消費電力の関係については，実行時間については CPU と比べて GPU の方が実行時間の増加度合いが小さく，反復回数が多くなっても実行時間が延びにくい傾向にあることがわかった．つづいて反復回数と消費エネルギーの関係については，消費電力については CPU は反復回数が増加すると消費電力が増加する傾向であるのに対して，GPU はほぼ一定 (横ばい) となっていることがわかった．これらの結果から，消費エネルギーについては反復回数が多い場合に GPU の方が有利となる (低くなる) ことが期待され，実際に消費エネルギーを確認してみたところ，500 反復周辺で消費エネルギーが逆転しており，反復回数が 500 回未満の場合には CPU の方が消費エネルギーが小さく，反復回数が 500 回以上の場合には GPU の方が消費エネルギーが小さくなった (図 6)．これらの結果から，対象問題の反復回数に応じて CPU と GPU から最適な実行ハードウェアを選択できることが確認できた．

リスト 2　自動生成されたコードの例

```
先頭に+印の行 : 電力関連API関連の記述
先頭に=印の行 : ATMathCoreLib関連の記述

  for(iloop_n=OAT_STARTTUNESIZE;
      iloop_n<=OAT_ENDTUNESIZE;iloop_n+=OAT_SAMPDIST){
    nstep = iloop_n;
    ...
=   exdesign_t exdes = new_exdesign(2, OAT_EPM_KAPPA);
=   for(iloop_idx=0;
        iloop_idx<OAT_EPM_MAXSAMP;iloop_idx++){
      double atmeter = 0.0;
+     double avtmp = P_getTemp();
=     int cand_idx=exdes_osa_atm(exdes,avtmp,&atmeter);
=     if (cand_idx >= 2) break;
                        // enough information obtained
      iusw1 = fpos[cand_idx];
      // --- power monitoring setup
+     P_initial();  P_start();
      t1 = OAT_Wtime();
      for(i=0; i<OAT_MAXREPEAT; i++) {
        OAT_InstallSelectPhase(
            mgn,nx,ny,n,dx,dy,delta,nstep,nout,
            calc_coef_a_w_pmobi,init,model,iusw1);
      }
      t2 = OAT_Wtime();
      t_all2 = (t2 - t1)/(double)OAT_MAXREPEAT;
      // --- power monitoring finalizing
+     P_stop();
      // --- get power data
+     num = P_recv(&rec);
+     P_recv(&rec);
+     P_close();
      if (num == 0) { // failed to get power
        ...
      }
      t_all = P_compPower(num, rec);
=     update_exdes(exdes,cand_idx,t_all2,t_all,avtmp);
=     iBestSw1 = getbest_exdes(exdes);
      assert(iBestSw1 >= 0);
      iBestSw1 = fpos[iBestSw1];
=     del_exdesign(exdes);
      ...
    }
  }
```

6. おわりに

本稿では著者らが CREST プロジェクト「ULP-HPC: 次世代テクノロジのモデル化・最適化による超低消費電力ハイパフォーマンスコンピューティング」において実施した超低消費電力ハイパフォーマンスコンピューティングの実現に向けた取り組みについて紹介した．また開発した機構を用いて電力情報を用いた最適化 (消費エネルギー最少化) の例を示した．これからの計算機システムの性能向上においては，省電力化はモバイル端末からスーパーコンピュータまでどのようなシステムにおいても非常に重要な課題である．我々も本プロジェクトにおける取り組みをさらに発展させて計算機システムの省電力化を進めていく予定である．

謝辞　本研究の一部は「ULP-HPC: 次世代テクノロジのモデル化・最適化による超低消費電力ハイパフォーマンスコンピューティング」(CREST 領域「情報システムの超低消費電力化を目指した技術革新と統合化技術」) の支援を受けています．

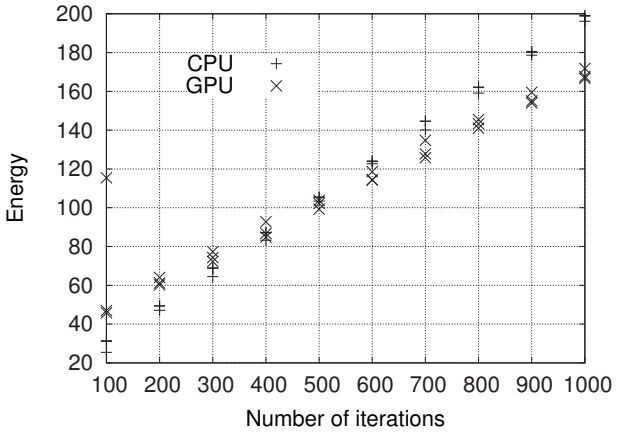

図 6 実験結果: 反復回数と消費エネルギー

参考文献

[1] 独立行政法人科学技術振興機構, ＜戦略的創造研究推進事業：CREST ＞情報システムの超低消費電力化を目指した技術革新と統合化技術, http://www.ulp.jst.go.jp/index.html

[2] RIKEN AICS, 京コンピュータ, http://www.aics.riken.jp/k/

[3] GPGPU.org, General-Purpose computation on Graphics Processing Units, http://gpgpu.org/

[4] NVIDIA, Developer Zone (CUDA ZONE), http://developer.nvidia.com/category/zone/cuda-zone

[5] OpenACC Home, http://www.openacc.org/

[6] Reiji Suda, Da Qi Ren: “Accurate Measurements and Precise Modeling of Power Dissipation of CUDA Kernels toward Power Optimized High Performance CPU-GPU Computing”, Proceedings of Workshop on Ultra Performance and Dependable Acceleration Systems (UPDAS), Hiroshima, pp.432-438 (2009).

[7] Cheng Luo, Kamil Rocki, Reiji Suda: “A precise measurement tool for power dissipation of CUDA kernels”, IPSJ SIG Technical Reports, Vol.2012-HPC-133 No.2, 第 133 回 HPC 研究会, 有馬ビューホテルうらら, March 26-27 (26) (2012).

[8] Satoshi OHSHIMA, Shoichi HIRASAWA, Hiroki HONDA: “OMPCUDA : OpenMP Execution Framework for CUDA Based on Omni OpenMP Compiler”, 6th International Workshop on OpenMP, Epochal Tsukuba, June 14-16 (2010).

[9] Seyong Lee, Rudolf Eigenmann: “OpenMPC: Extended OpenMP Programming and Tuning for GPUs”, SC10: Proceedings of the 2010 ACM/IEEE conference on Supercomputing (2010).

[10] Takahiro Katagiri, Kenji Kise, Hiroki Honda, Toshitsugu Yuba: “ABCLibScript: A Directive to Support Specification of An Auto-tuning Facility for Numerical Software”, Parallel Computing, Vol.32, Issue 1, pp.92-112 (2006).

[11] 須田礼仁: “自動チューニング数理基盤ライブラリ ATMathCoreLib”, 情報処理学会 研究報告 HPC-129-14 (2011).