Vol.2012-HPC-135 No.18
2012/8/2

省電力化を意識した 線形数値計算ライブラリの実装と評価

田中献大† 黒田久泰†

本稿では，計算性能と消費電力情報に基づき消費電力を最適化する線形数値計算ライブラリの実装と評価について述べる．実装については，BLAS レベル 1 相当の基本的な数値計算を行うライブラリを SIMD 命令である SSE, AVX 命令，OpenMP を用いて記述した．そして，CPU 動作周波数，スレッド数を変更することで 1W あたりの計算性能が高くなることを実験により明らかにし，消費電力測定ライブラリを通し消費電力情報を取得し 1W あたりの計算性能の高い組み合わせを自動で選択する機構を実装し評価を行った．

Implementation and Evaluation of Power-aware Linear Algebra Library

Kenta Tanaka† Hisayasu Kuroda†

This paper describes the implementation and evaluation of the linear algebra library to optimize the power consumption based on the computational performance and the power consumption information. We have developed the basic linear algebra library equivalent of BLAS level 1 by use of SSE instruction, AVX instruction, and OpenMP directive. Evaluations show that an appropriate setting of CPU frequency and the number of execution threads improves computational performance per watt. In addition, we have developed a measurement program for power consumption and a mechanism for selecting the best combination to get the highest computational performance per watt.

1. はじめに

近年，計算機システムの高性能化や電力供給不足に伴い，省電力化はとても重要な課題になってきている．そこで，我々は省電力化を意識した線形数値計算ライブラリの開発を行っている．

一般に，CPU 周波数が高ければ高いほど計算性能は向上するが，計算性能の向上に比べて電力消費量の向上の方が上回ってしまうことの方が多い．また，実行スレッド数についても，通常であれば，論理 CPU コア数と同じスレッド数で実行すれば一番高い性能が得られることになるが，その計算性能の向上よりも電力消費量の向上の方が上回ってしまうことがある．このような場合，計算性能の低下を許容することで，省電力化を実現することが可能である．

我々は，計算性能と消費電力を測定するプログラムを用いて消費電力情報を作成し，線形数値計算ライブラリ内で消費電力情報を元に CPU 動作周波数, 実行スレッド数，実装方式の 3 つを動的に変更して，1W あたりの計算性能を最大化する方法を提案する．

本論文では，2 章で関連研究について述べ，3 章において我々の開発した線形数値計算ライブラリの構成について述べ，4 章では，実際の計算機システムにおける測定結果について述べる．5 章では，共役勾配法に適用した場合について評価し，6 章でまとめを行う．

† 愛媛大学
Ehime University

2. 関連研究

近年の CPU の多くに動的電源電圧・周波数制御機構である DVFS(Dynamic Voltage and Frequency Scaling)が備わっており，CPU の電圧および周波数を変更することができる．一般的な CMOS により構成される CPU は動作周波数と動作電圧の二乗に比例するため，電圧および周波数を変更することは CPU の消費電力を削減するうえで非常に効果的である[1]．更に，メモリの電圧や周波数を変更し消費電力を下げる先行研究[2][3]が存在する．

DVFS を活用するシステムの一つとして Linux の ondemand governor がある．これは CPU 利用率に応じて，CPU 周波数を設定する仕組みであり，システムの CPU 使用率が低い場合に動作周波数を低く設定することで省電力化する．パフォーマンスカウンタである PMC(Performance Monitor Controller)を用いて独自のスケジューリングを行う先行研究[4]も存在する．

CPU 周波数を制御する手法は，事前にアプリケーションの特徴を分析し CPU 周波数を決定するオフライン手法[5]，PMC などの情報により事前の分析なしに CPU 周波数を決定するオンライン手法[6][7]が知られており，我々のライブラリはオフライン手法に属する．

しかし，本研究では，プロセス自身が実行される CPU 周波数を決定しているという点で大きく異なる．そのため，対象とする Linux システムの governor の設定値は userspace として，周波数を設定するデバイスファイル scaling_setspeed への書き込み権限が必要である．

3. 線形数値計算ライブラリの構成

我々の開発した数値計算ライブラリは，計算機の省電力化を目的としており，大きく分けて，(1)線形基本演算ライブラリ，(2)計算性能と消費電力測定プログラム，(3)線形応用演算ライブラリの 3 つの部分から構成される．x86-64 CPU を搭載した Linux システムを対象としており C 言語で記述されている．

3.1 線形基本演算ライブラリ

我々はベクトルや行列などの線形基本演算を行うライブラリを独自に開発している．ベクトル同士の演算を主とする BLAS(Basic Linear Algebra Subprograms)レベル 1 に相当するルーチンとして，表 1 に示す 11 個が用意されている．また，各ルーチンに，組み込み関数(intrinsic)を用いずに C 言語で記述したもの，SSE(Streaming SIMD Extensions)命令を用いたもの，AVX(Intel Advanced Vector eXtentions)命令を用いたものの 3 種類を用意している．

SSE 命令は Intel Pentium4 に搭載された SIMD 命令であり，専用の XMM レジスタ(32bit モード: 8 個，64bit モード: 16 個)に格納された 128bit のデータに対し SIMD 演算が可能である．我々が開発したライブラリでは，SSE2 を intrinsic 文で記述している．このため，x86-64 アーキテクチャの全ての CPU で実行可能である．

AVX 命令は 2011 年 1 月に発表された Intel 第 2 世代 Core i7(コードネーム: Sandy Bridge)に搭載された SIMD 命令であり，専用の YMM レジスタ(32bit モード: 8 個，64bit モード: 16 個)に格納された 256bit のデータに対し SIMD 演算が可能である．AVX 命令は SSE 命令を混在させるとペナルティが発生し大幅に計算性能が低下するため注意が必要である．我々の開発したライブラリでは，AVX についても intrinsic 文を用いて記述している．並列化は OpenMP を用いている．

表 1　基本演算ライブラリのルーチン一覧
Table 1　Routine of our linear algebra library

dset(n, s, x)	$x_i = s$
dsum(n, x)	$\sum x_i$
ddot(n, x, y)	$(\mathbf{x}, \mathbf{y})$
ddot2(n, x)	$(\mathbf{x}, \mathbf{x})$
dcopy(n, x, y)	$\mathbf{y} = \mathbf{x}$
dscal(n, s, x)	$\mathbf{x} = s * \mathbf{x}$
dscaly(n, s, x, y)	$\mathbf{y} = s * \mathbf{x}$
dadd(n, x, y)	$\mathbf{y} = \mathbf{x} + \mathbf{y}$
daxpy(n, s, x, y)	$\mathbf{y} = s * \mathbf{x} + \mathbf{y}$
daxpyx(n, s, x, y)	$\mathbf{x} = s * \mathbf{x} + \mathbf{y}$
daxpyz(n, s, x, y, z)	$\mathbf{z} = s * \mathbf{x} + \mathbf{y}$

n はベクトルのサイズ

3.2 計算性能と電力測定プログラム

計算性能と電力測定プログラムの開発にあたり，効率良く消費電力を測定するために消費電力測定ライブラリを作成した．このライブラリは RS-232C や GPIB(IEEE488.2)のインターフェースを持つ電力測定装置を制御し電力情報を取得することを目的としている．

このライブラリは電力測定装置の制御や電力測定装置から測定データを受け取り TCP 通信により測定データをクライアントに配信するサーバープログラムと，クライアントプログラムから構成されている．クライアントプログラムでは，表 2 に示す 3 つの API が存在し，これらの API を用いて電力測定装置からの測定データを受け取ることができる．

表 2　クライアント側で使用する電力測定 API
Table 2　Power measurement API for client

pm_start	電力測定を開始する
pm_end0	電力測定を終了し 簡易測定データを取得する
pm_end1	電力測定を終了し 詳細測定データを取得する

電力測定装置との直接のやりとりを行うサーバープログラムとサーバープログラムから電力測定結果を受け取るクライアントプログラムに分けることで，できるだけ正確に計算性能と電力消費量が測定できるようにした．

3.3 線形応用演算ライブラリ

我々の開発したライブラリでは各関数の実行を最適化するために消費電力情報を必要とする．我々の開発したライブラリではビルド時に下記の消費電力測定ライブラリ，電力装置を使用し命令実行時の消費電力，計算性能を測定し消費電力情報とする．

我々の開発したライブラリでは計算機の対応する命令セット，消費電力情報，ベクトル長を元に命令，スレッド数，CPU 周波数を変更することで計算機の消費電力を削減している．我々の開発したライブラリでは最適な命令セット, スレッド数を自動的に選択して計算を行うが，表 3 に示す電力制御 API が用意されており，明示的に指定することも可能である．実行時にスレッド数，CPU 周波数を基本的に最も計算性能が高い組み合わせを選択するが，計算性能低下許容度が設定されている場合，上の命令の計算性能から許容された組み合わせのうち最も 1W あたりの計算性能が高い組み合わせを選択する．すなわち，計算性能低下許容量を指定しなければ最も計算性能が速い組み合わせで実行され，処理低下許容量が指定されれば許容量が満たされる範囲で 1W あたりの計算性能が最大となるように設定される．

表 3 電力制御 API 一覧
Table 3　Power control API

set_slowdown	計算性能低下許容度の指定
set_instruction	命令セットの指定
adjust_freq_by_func	関数単位での周波数制御
focus_on_function	最適化する関数の指定

4. 性能評価

今回実装したライブラリの性能を評価するために，

- C での一般的な実装(Normal)
- SSE 命令を使用した実装(SSE)
- AVX 命令を使用した実装(AVX)

の計算性能，平均電力量を比較し 1W あたりの計算性能が高くなる組み合わせを明らかにする．

ライブラリの性能の測定は表 4 の環境で行った．Core i7 3930K には TB(Intel Turbo Boost Technology)が搭載されており，CPU の発熱量に余裕がある場合，定格動作周波数である 3.2GHz 以上の周波数に自動的にオーバークロックを行い計算性能を向上させる機構がある[1]．TB 機構は CPU 温度により上昇する周波数が前後してしまうため，CPU クーラーのファン回転数を最大とし，またマザーボード周辺を送風機で冷却することで TB が許容する最大周波数で長時間動作できるようにしている．

表 4 計算機環境
Table 4 Computer environment

CPU	Intel Core i7 3930K 3.20GHz
Motherboard	ASRock X79 Extreme9
Memory	合計 64GB, DDR3-1333 AD3U1333W8G9-2 を 4 セット
SSD	PLEXTOR PX-128M3P
Video card	Giada G210-LP (GeForce 210 DDR3 512MB)
Power supply	SilverStone SST-ST75F-G-E
OS	Fedora 16 (Linux 3.4.2-1 SMP x86_64)
コンパイラ	gcc 4.6.3
コンパイル オプション	SSE ルーチン: -O3 -msse2 –fopenmp それ以外: -O3 -mavx –fopenmp
電力測定装置	日置 AC/DC パワーハイテスタ 3334-01

消費電力測定には表 4 の計算機とは異なる計算機に電力測定サーバープログラムを動作させ，電力測定装置として日置 AC/DC パワーハイテスタ 3334-01 を GPIB インターフェースで接続し，表 4 の計算機を電力測定装置に接続して測定を行った．これは電力測定装置制御のために計算性能や消費電力への影響を防ぐためである．電力測定装置は電圧レンジ 150V, 電流レンジ 3A とした．日置 AC/DC パワーハイテスタ 3334 では 0.2 秒の測定間隔で有効電力を測定できる．

計算性能，平均電力量，1W あたりの計算性能を明らかにするため，dsum, daxpy を以下の組み合わせで 10 秒以上反復実行させ，計算性能，平均電力量を測定し，1W あたりの計算性能を求めた．

- ベクトルのサイズは 920, 1,920, 3,840, 7,680, 15,360, 30,720, 61,440, 122,880, 245,760, 491,520, 983,040, 1,966,080, 3,932,160, 7,864,320
- 実行スレッド数は 1, 2, 3, 4, 5, 6
- CPU 周波数は 1.2GHz から 3.8GHz まで 0.2GHz 刻み

また, 計算性能の単位は GFlops, 平均電力量の単位は W, 1W あたりの計算性能の単位は$\times 10^{-3}$GFlops/W とする．なお, 実行スレッド数を 12 とした場合, ほとんどの場合で 6 スレッドに比べて計算性能が減少するにも関わらず消費電力量が上がるため測定対象から外した．

測定数が膨大なため，上記条件で典型的な例である，命令セット Normal, SSE, AVX，サイズ 960, 30,720, 983,040, 7,864,320，実行スレッド数 1, 3, 6，CPU 周波数 2.6GHz, 3.0GHz, 3.2GHz と変化させた dsum の計算性能, 平均電力量, 1W あたりの計算性能を表 5，daxpy の計算性能, 平均電力量, 1W あたりの計算性能を表 6 に示す．

まず, 計算性能について着目すると表 5 と表 6 より次のことが言える．

- AVX は Normal と比較して平均 2.49 倍計算性能が高い．
- dsum の計算性能は AVX が SSE と比較して平均 1.05 倍高く，daxpy の計算性能は AVX が SSE と比較して平均 1.10 倍高い．
- dsum の計算性能は最大 23.02GFlops(理論性能の 27%)であり，daxpy の計算性能は最大 15.80GFlops(理論性能の 19%)である．
- サイズが小さいとき, 6 スレッドで計算性能が低いのは，スレッド生成にかかるオーバーヘッドが実計算より多く占めるためである．
- 周波数を 2.6GHz に設定すると，3.0GHz や 3.2GHz と比較して周波数差以上に計算性能が低下するが，これは CPU のメモリコントローラがメモリのチャネル数，または周波数を制御している可能性がある．

次に，平均電力量と 1W あたりの計算性能について着目すると表 5 と表 6 より次のことが言える．

- AVX は Normal と比較して平均 2.50 倍 1W あたりの計算性能が高い．
- 周波数を 3.0GHz に設定したときは周波数を 3.2GHz に設定したときと比較して平均 1.02 倍 1W あたりの計算性能が高い．
- dsum の 1W あたりの計算性能は AVX が SSE と比較して平均 1.02 倍高く, daxpy の 1W あたりの計算性能は AVX が SSE と比較して平均 1.03 倍高い．
- どの実装でも消費電力は±5%程度しか変わらない．

よって, AVX 命令を使用すると十分な計算性能が得られ, CPU 周波数，スレッド数を適切に選択すると効果的に消費電力を減少できることを確認した．

表 5 と表 6 に相当する消費電力情報を元に計算性能が最も高い組み合わせ，最高性能の 90, 80, 70%の計算性

1 1~2 スレッド実行時:3.8GHz，3 スレッド実行時:3.7GHz，4 スレッド実行時: 3.6GHz，5~6 スレッド実行時:3.5GHz までオーバークロックする．

能低下を許容できる場合の 1W あたりの計算性能が最も高い組み合わせを自動的に選択する機構を実装し，実際に選択された命令，CPU 周波数，スレッド数と最高周波数で 6 スレッドでの計算性能と 1W あたりの計算性能を表 7, 表 8 に示す. また, 1W あたりの計算性能を図 1, 図 2 に示す．1W あたりの計算性能が高いほど省電力であると判断する．全てのサイズ，性能で AVX が使用されているが，これは AVX 命令と SSE 命令を混在すると発生するペナルティを避けるため AVX 命令が使用可能な CPU の場合，SSE 命令が使用されないためである．

表 7 と表 8 より次のことが言える．

- dsum の 1W あたりの計算性能は最高性能が最高周波数で 6 スレッドと比較して平均 1.70 倍高く，daxpy の 1W あたりの計算性能は最高性能が最高周波数で 6 スレッドと比較して平均 1.67 倍高い．
- dsum の 1W あたりの計算性能は性能 70%が最高性能と比較して平均 1.24 倍高く, daxpy の 1W あたりの計算性能は性能 70%が最高性能と比較して平均 1.09 倍高い．
- サイズが小さいとき TB が使用されている．
- 選択されたスレッド数，周波数から Core i7 3930K ではサイズが小さいとき 1~2 スレッドで 3.2GHz, L3 に収まるサイズのとき 6 スレッドで 3.0GHz, L3 に収まらないサイズの時 2~4 スレッドで 3.0GHz が基本的に最適であると判断されているが，この結果は一般的に効率の悪いと言われているスレッド数が少ない、周波数が高い組み合わせを省くことで高速に消費電力情報を構築することが難しいことを意味する。

また，図 1 と図 2 より次のことが言える．

- daxpy は dsum と比較して L3 キャッシュの効果が少なくサイズ 491,520 と 983,040 のときの性能が低い．
- サイズが小さいとき, 性能を低下させても 1W あたりの計算性能はほとんど向上しない．

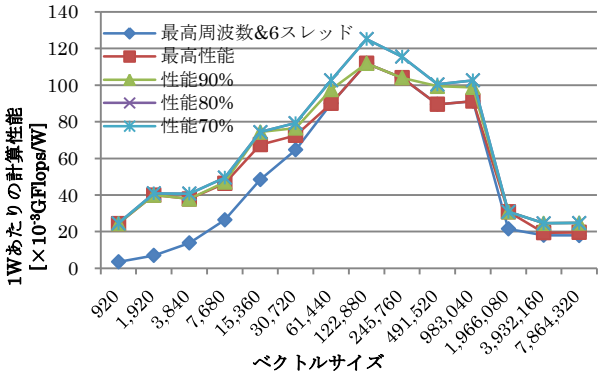

図 1　dsum の 1W あたりの計算性能
Figure 1　Computational performance per watt in dsum

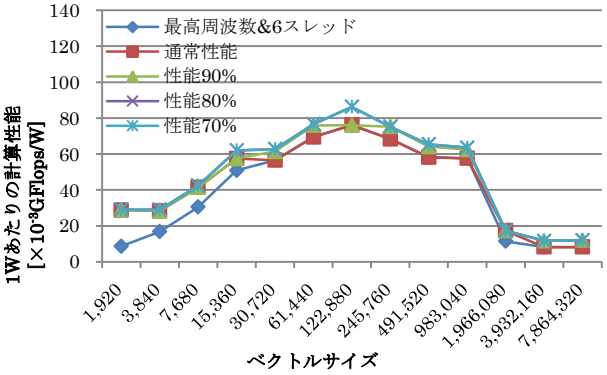

図 2　daxpy の 1W あたりの計算性能
Figure 2　Computational performance per watt in daxpy

表 5　dsum の計算性能，平均電力量，1W あたりの計算性能
Table 5　Computational performance and average power consumption in dsum

サイズ		960			30,720			983,040			7,864,320		
周波数	コア	Normal	SSE	AVX	Normal	SSE	AVX	Normal	SSE	AVX	Normal	SSE	AVX
2.6 GHz	1	0.421	1.192	1.453	0.532	2.121	2.196	0.539	1.774	1.769	0.480	1.065	1.062
		92.96	95.43	96.40	93.62	95.67	96.20	94.32	96.16	100.91	106.54	113.50	114.52
		4.527	12.491	15.071	5.687	22.169	22.821	5.712	18.452	17.535	4.505	9.386	9.275
	3	0.378	0.501	1.017	1.459	5.267	5.379	1.599	5.197	5.172	1.397	2.452	2.466
		109.22	107.51	109.19	108.86	112.63	113.25	110.82	122.84	118.92	133.19	143.20	147.92
		3.464	4.655	9.313	13.402	46.764	47.490	14.425	42.304	43.486	10.490	17.124	16.673
	6	0.312	0.303	0.328	2.458	6.157	6.348	3.160	10.283	10.083	2.519	2.765	2.788
		127.61	124.79	128.53	130.46	131.69	134.73	134.69	147.36	149.00	166.10	166.30	170.98
		2.447	2.426	2.553	18.839	46.751	47.120	23.459	69.785	67.668	15.167	16.624	16.304
3.0 GHz	1	0.728	2.071	2.506	0.922	3.530	3.872	0.933	3.070	3.062	0.831	1.730	1.752
		102.82	102.38	103.74	100.99	103.39	103.38	101.06	107.38	107.77	112.20	120.32	122.98
		7.084	20.226	24.153	9.131	34.138	37.451	9.229	28.586	28.416	7.402	14.380	14.247
	3	0.710	0.835	0.897	2.544	9.171	9.232	2.777	9.118	8.948	2.376	3.829	3.846
		120.68	118.96	119.31	119.49	124.67	124.38	121.39	128.96	133.68	143.26	154.42	157.73
		5.886	7.019	7.517	21.293	73.566	74.221	22.878	70.703	66.934	16.586	24.794	24.382
	6	0.540	0.546	0.553	4.265	10.977	11.053	5.460	17.773	17.447	4.004	4.230	4.251
		144.36	139.72	143.02	147.85	149.04	152.55	155.67	166.79	169.88	182.23	181.50	189.44
		3.738	3.910	3.869	28.844	73.650	72.457	35.070	106.558	102.700	21.969	23.307	22.437
3.2 GHz	1	0.778	2.198	2.674	0.984	4.310	4.266	0.995	3.275	3.264	0.877	1.801	1.833
		104.97	107.79	108.91	104.47	107.51	107.29	105.68	108.68	109.53	116.61	128.83	127.49
		7.413	20.392	24.548	9.419	40.088	39.764	9.414	30.134	29.798	7.520	13.977	14.377
	3	1.158	0.913	0.921	2.707	9.726	9.965	2.957	9.396	9.521	2.509	3.856	3.908
		122.04	122.91	125.80	125.25	129.22	130.56	130.83	141.70	141.08	158.35	164.58	170.47
		9.487	7.429	7.318	21.615	75.272	76.319	22.601	66.306	67.486	15.844	23.430	22.924
	6	0.571	0.575	0.573	4.517	11.557	11.463	5.833	18.976	18.605	4.037	4.245	4.255
		143.17	148.58	152.03	157.85	159.25	163.99	167.48	181.54	188.26	195.35	193.16	202.55
		3.985	3.867	3.768	28.617	72.572	69.903	34.829	104.530	98.826	20.667	21.976	21.009

上段:計算性能[GFlops], 中段:平均電力量[W], 下段:1W あたりの計算性能[$\times 10^{-3}$GFlops/W]

表 6　daxpy の計算性能，平均電力量，1W あたりの計算性能
Table 6　Computational performance and average power consumption in daxpy

サイズ		960			30,720			983,040			7,864,320		
周波数	コア	Normal	SSE	AVX	Normal	SSE	AVX	Normal	SSE	AVX	Normal	SSE	AVX
2.6 GHz	1	0.539	0.964	1.191	0.558	1.381	1.381	0.435	1.044	1.128	0.420	0.704	0.728
		97.63	95.73	97.24	98.14	95.26	96.07	116.42	99.81	100.49	124.58	122.30	119.32
		5.522	10.065	12.252	5.682	14.491	14.374	3.740	10.460	11.226	3.370	5.760	6.099
	3	0.444	0.537	0.531	1.810	3.691	3.818	1.027	3.054	3.326	0.813	1.240	1.228
		110.29	108.07	111.49	118.07	113.40	114.17	149.40	119.47	122.01	154.72	150.25	150.39
		4.022	0.497	0.476	1.533	3.255	3.344	0.688	2.556	2.726	0.525	0.825	0.816
	6	0.360	0.401	0.394	3.102	5.412	5.583	1.197	5.883	6.476	0.839	1.302	1.278
		130.20	125.59	129.25	143.70	134.13	136.48	172.41	147.69	151.81	180.31	167.72	173.57
		2.761	3.193	3.049	21.588	40.345	40.907	6.942	39.832	42.657	4.653	7.765	7.362
3.0 GHz	1	0.931	1.711	2.062	0.966	2.350	2.590	0.733	1.816	1.983	0.682	1.136	1.172
		103.51	102.35	106.53	109.44	102.61	103.17	123.07	107.32	105.51	127.70	125.00	125.97
		8.993	16.717	19.353	8.823	22.904	25.105	5.953	16.917	18.791	5.338	9.088	9.303
	3	0.757	0.929	0.928	3.231	6.381	6.627	1.577	5.288	5.752	1.234	1.839	1.857
		122.95	119.47	120.86	129.62	123.95	124.81	159.66	128.12	133.77	160.42	154.07	157.90
		6.154	7.774	7.676	24.924	51.483	53.097	9.878	41.277	42.996	7.690	11.935	11.760
	6	0.605	0.695	0.669	5.345	9.443	9.575	1.797	10.154	11.055	1.264	1.962	1.955
		142.98	141.15	145.29	164.62	152.41	155.36	190.54	168.67	173.78	189.97	181.12	185.84
		4.233	4.924	4.602	32.466	61.955	61.632	9.431	60.202	63.611	6.652	10.830	10.520
3.2 GHz	1	1.002	1.823	0.275	1.027	2.406	2.645	0.759	1.933	2.115	0.715	1.178	1.212
		111.61	108.20	106.75	111.20	107.64	108.18	129.84	112.19	109.61	137.38	129.88	135.77
		8.981	16.850	2.575	9.234	22.352	24.452	5.846	17.226	19.296	5.202	9.071	8.923
	3	0.773	0.989	0.956	3.445	6.832	7.011	1.553	5.623	6.120	1.244	1.863	1.890
		129.72	122.98	126.28	137.37	129.29	131.59	167.34	137.69	138.44	170.73	163.55	166.96
		5.959	8.041	7.570	25.080	52.846	53.277	9.280	40.837	44.203	7.286	11.393	11.320
	6	0.660	0.732	0.733	5.709	10.028	10.307	5.916	10.894	11.743	1.274	1.938	1.915
		151.47	150.27	156.03	179.68	162.95	167.88	204.97	176.93	188.04	211.12	195.60	197.35
		4.357	4.873	4.700	31.775	61.541	61.393	28.861	61.569	62.446	6.033	9.909	9.703

上段:計算性能[GFlops]，中段:平均電力量[W]，下段: 1W あたりの計算性能[$\times 10^{-3}$GFlops/W]

表 7　dsum における 1W あたりの計算性能と最適な実装方式
Table 7　Computational performance per watt in dsum

サイズ	960	1,920	3,840	7,680	15,360	30,720	61,440	122,880	245,760	491,520	983,040	1,966,080	3,932,160	7,864,320
最高性能	3.19	5.17	4.90	7.05	11.14	12.91	18.04	23.02	22.48	19.81	20.35	5.56	4.29	4.33
	24.40	39.98	37.86	46.30	67.50	72.54	90.02	111.93	104.01	89.48	91.29	30.85	19.42	19.61
	A-1	A-1	A-1	A-3	A-4	A-5	A-6	A-6	A-6	A-6	A-6	A-5	A-5	A-5
	3.8GHz	3.8GHz	3.8GHz	3.7GHz	3.6GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.0GHz	3.5GHz	3.5GHz
最高周波数 6 スレッド	0.63	1.27	2.50	4.88	9.07	12.62	18.04	23.02	22.48	19.81	20.35	5.09	4.28	4.28
	3.47	7.00	13.77	26.35	48.44	64.70	90.02	111.93	104.01	89.48	91.29	21.49	17.99	17.88
	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6
	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz
性能 90%	3.19	5.17	4.90	6.81	10.26	11.73	16.38	23.02	22.48	18.25	18.61	5.56	4.18	4.20
	24.40	39.98	37.86	47.05	74.42	76.61	97.69	111.93	104.01	99.39	98.83	30.85	24.56	24.75
	A-1	A-1	A-1	A-2	A-4	A-5	A-6	A-6	A-6	A-6	A-6	A-5	A-4	A-4
	3.8GHz	3.8GHz	3.8GHz	3.8GHz	3.2GHz	3.2GHz	3.2GHz	3.5GHz	3.5GHz	3.2GHz	3.2GHz	3.0GHz	3.0GHz	3.0GHz
性能 80%	2.67	4.35	4.34	6.25	同上	11.17	14.95	19.90	19.16	17.00	17.45	同上		
	24.55	40.93	40.73	49.50		79.27	102.73	125.16	115.56	100.44	102.65			
	A-1	A-1	A-1	A-3		A-4	A-6	A-6	A-6	A-6	A-6			
	3.2GHz	3.2GHz	3.2GHz	3.2GHz		3.2GHz	3.0GHz	3.0GHz	3.0GHz	3.0GHz	3.0GHz			
性能 70%	同上													

1 段目:計算性能[GFlops]，2 段目:1W あたりの計算性能[$\times 10^{-3}$GFlops/W]，3 段目:命令セット(N:Normal, S:SSE, A:AVX)-スレッド数，4 段目:周波数

表 8　daxpy における 1W あたりの計算性能と最適な実装方式
Table 8　Computational performance per watt in daxpy

サイズ	960	1,920	3,840	7,680	15,360	30,720	61,440	122,880	245,760	491,520	983,040	1,966,080	3,932,160	7,864,320
最高性能	2.62	3.72	4.07	6.38	9.99	11.24	14.12	15.80	14.77	12.89	13.02	2.96	1.98	1.97
	19.71	28.83	28.28	41.35	57.46	56.49	69.45	76.06	68.37	58.21	57.62	17.30	9.39	10.47
	A-1	A-1	A-2	A-3	A-5	A-6	A-6	A-6	A-6	A-6	A-6	A-5	A-5	A-5
	3.8GHz	3.8GHz	3.8GHz	3.7GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.2GHz	3.5GHz	3.2GHz
最高周波数 6 スレッド	0.81	1.59	3.08	5.69	9.68	11.24	14.12	15.80	14.77	12.89	13.02	2.60	1.88	1.94
	4.45	8.70	16.77	30.52	50.91	56.49	69.45	76.06	68.37	58.21	57.62	11.49	8.24	8.28
	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6	A-6
	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz	3.5GHz
性能 90%	2.62	3.72	4.07	5.73	8.82	10.31	12.98	15.80	13.54	11.76	11.74	2.68	1.97	1.96
	19.71	28.83	28.28	41.35	60.11	61.39	75.97	76.06	75.18	64.06	62.45	17.30	11.76	11.61
	A-1	A-1	A-2	A-3	A-5	A-6	A-6	A-6	A-6	A-6	A-6	A-3	A-4	A-4
	3.8GHz	3.5GHz	3.8GHz	3.7GHz	3.5GHz	3.2GHz	3.2GHz	3.5GHz	3.2GHz	3.2GHz	3.2GHz	3.0GHz	3.0GHz	3.0GHz
性能 80%	2.20	3.09	3.42	5.14	8.78	9.14	12.12	13.85	同上	11.08	11.05	同上		1.76
	20.60	28.99	29.11	42.40	62.08	62.76	76.61	86.25		65.41	63.58			11.99
	A-1	A-1	A-2	A-3	A-5	A-5	A-6	A-6		A-6	A-6			A-2
	3.2GHz	3.2GHz	3.2GHz	3.0GHz	3.0GHz	3.0GHz	3.0GHz	3.0GHz		3.0GHz	3.0GHz			3.0GHz
性能 70%	同上													

1 段目:計算性能[GFlops]，2 段目:1W あたりの計算性能[$\times 10^{-3}$GFlops/W]，3 段目:命令セット(N:Normal, S:SSE, A:AVX)-スレッド数，4 段目:周波数

表 9　共役勾配法に適応したときの計算性能
Table 9　Computational performance in CG method

サイズ	30,720				7,864,320			
	daxpy	ddot	ddot2	向上比	daxpy	ddot	ddot2	向上比
最高性能	11.24 56.49 A-6/3.5GHz	9.26 45.59 A-6/3.5GHz	12.77 65.97 A-6/3.5GHz	0% 0%	1.94 8.28 A-6/3.5GHz	2.10 9.94 A-4/3.6GHz	4.32 20.29 A-6/3.5GHz	0% 0%
最高周波数 6 スレッド	11.24 56.49 A-6/3.5GHz	9.26 45.59 A-6/3.5GHz	12.77 65.97 A-6/3.5GHz	0% 0%	1.94 8.28 A-6/3.5GHz	2.07 8.51 A-6/3.5GHz	4.30 18.82 A-6/3.5GHz	−1% −17%
性能 90%	10.31 61.39 A-6/3.2GHz	8.35 48.91 A-6/3.2GHz	11.87 76.42 A-5/3.2GHz	−8% 10%	1.96 11.61 A-4/3.0GHz	1.97 12.24 A-3/3.0GHz	4.21 24.44 A-4/3.0GHz	−2% 15%
性能 80%	9.14 62.76 A-5/3.0GHz	8.06 51.06 A-5/3.2GHz	11.87 76.42 A-5/3.2GHz	−15% 12%	1.76 11.99 A-2/3.0GHz	1.97 12.24 A-3/3.0GHz	4.21 24.44 A-4/3.0GHz	−8% 17%
性能 70%	同上				同上			
最高周波数 6 スレッド Normal	6.26 29.21 N-6/3.5GHz	4.95 24.80 N-6/3.5GHz	4.92 27.21 N-6/3.5GHz	−48% −50%	1.27 5.20 N-6/3.5GHz	2.05 8.75 N-6/3.5GHz	4.07 18.34 N-6/3.5GHz	−23% −34%

上段:計算性能[GFlops], 中段:1W あたりの計算性能[$\times 10^{-3}$GFlops/W], 下段:命令セット(N:Normal, S:SSE, A:AVX)-スレッド数/周波数

5. 共役勾配法への適応および評価

疎行列反復解法において，高橋秀俊版の共役勾配法[8]を我々の開発したライブラリにより実装した．一般的な共役勾配法では，$\mathbf{p} = \mathbf{r} + \beta\mathbf{p}$の演算が出てくるが，高橋秀俊版の共役勾配法では，$\mathbf{p} = \mathbf{p} + \beta\mathbf{r}$の演算になるため表 1 に示した daxpy 演算が利用できるという利点がある．$\mathbf{Ax} = \mathbf{b}$を解く高橋秀俊版の共役勾配法のアルゴリズムは

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;　$\beta_0 = \dfrac{1}{(\mathbf{r}_0, \mathbf{r}_0)}$;　$\mathbf{p}_0 = \beta_0 \mathbf{r}_0$

for $k = 0,1,\cdots$　until $\|\mathbf{r}_k\| \le \varepsilon \|\mathbf{b}\|$　do

$\alpha_k = \dfrac{1}{(\mathbf{p}_k, \mathbf{A}\mathbf{p}_k)}$

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$

$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$

$\beta_{k+1} = \dfrac{1}{(\mathbf{r}_{k+1}, \mathbf{r}_{k+1})}$

$\mathbf{p}_{k+1} = \mathbf{p}_k + \beta_{k+1}\mathbf{r}_{k+1}$

end

となる．疎行列ベクトル積の部分を除くと，1 反復あたり daxpy ルーチンが 3 回，ddot ルーチンが 1 回，ddot2 ルーチンが 1 回実行される．これらのことを考慮すると，共役勾配法を疎行列ベクトル積の部分を除き再現できる．疎行列 $\mathbf{A}$，各ベクトル$\mathbf{p}_k, \mathbf{A}\mathbf{p}_k, \mathbf{x}_k, \mathbf{r}_k$が十分に L3 キャッシュに収まると思われるサイズ 30,720 とキャッシュに収まらないサイズ 7,864,320 のベクトルに対し daxpy，ddot，ddot2 を行ったときに選択された命令セット，CPU 周波数，スレッド数，計算性能，1W あたりの計算性能の向上比を表 9 に示す．

表 9 より次のことが言える．

- サイズが 30,720 のとき，20%の処理性能低下を許容すれば処理性能低下 15%で 1W あたりの計算性能が 12%向上する．
- サイズが 7,864,320 のとき 20%の処理速度低下を許容すれば処理速度低下 8%で 1W あたりの計算性能が 17%向上する．

6. まとめ

我々は，消費電力情報に基づき消費電力を最適化する線形数値計算ライブラリの実装について述べた．共役勾配法に適応した場合，疎行列ベクトル積の部分を除くと 20%の計算性能低下を許容すれば処理速度低下 8%で最大 17%消費電力量を削減できることを示した．

今後の課題として，電力測定装置のない計算機を対象として多くの計算機に備わる温度センサーの示す値に基づいた消費電力推定を行う，ルーチンを増やす，計算性能の向上が挙げられる．

参考文献

1) 船岡健司, 加藤真平, 山崎信行: マルチプロセッサ用の実時間電圧周波数制御, 情報処理学会論文誌コンピューティングシステム, Vol.1, No.2, pp.96-110 (2008)
2) Qingyuan Deng, David Meisner, Luiz Ramos, Thomas F. Wenisch, and Ricardo Bianchini: MemScale: Active Low-Power Modes for Main Memory, ACM SIGARCH Computer Architecture News, Vol.39, No.1, pp.225-238 (2011)
3) Howard Davidy, Chris Fallinx, Eugene Gorbatovy, Ulf R. Hanebuttey, and Onur Mutlu: Memory Power Management via Dynamic Voltage/Frequency Scaling, Proceedings of the 8th ACM international conference on Autonomic computing, pp.31-40 (2011)
4) 金井遵, 佐々木広, 近藤正章, 中村宏, 天野英晴, 宇佐美公良, 並木美太郎: 性能予測モデルの学習と実行時性能最適化機構を有する省電力化スケジューラ, 情報処理学会論文誌コンピューティングシステム, Vol.49, No.SIG 2(ACS 21), pp.20-36 (2008)
5) 佐々木広，浅井雅司，池田佳路，近藤正章，中村宏: 統計情報に基づく動的電源電圧制御手法, 情報処理学会論文誌コンピューティングシステム，Vol.47，No.Sig18 (ACS16)，pp.80-91 (2006)
6) 近藤正章，中村宏: 主記憶アクセスの負荷情報を利用した動的周波数変更による低消費電力化, 情報処理学会論文誌コンピューティングシステム, Vol.45, No.SIG 6(ACS 6), pp.1-11 (2004)
7) 三輪真弘，中島耕太，平井聡，成瀬彰: メモリ消費電力に基づく CPU 周波数の動的制御, 情報処理学会研究報告, Vol.2011-HPC-130, No.24, pp.1-7 (2011)
8) 戸川隼人: 共役勾配法，シリーズ新しい応用の数学 17，教育出版(1977)