大規模並列システムにおける電力最適化時の消費エネルギー評価

木村英明[†,††] 今田貴之[†] 佐藤三久[†]

近年，PC クラスタに代表される大規模並列システムを構築する上でシステムの省電力性が重要になっている．高性能計算向けアプリケーションではプログラムを複数領域に分割し動作 P-State を制御する消費エネルギー最適化が有効であるが，大規模システムでの消費エネルギー最適化効果や消費エネルギー最適化に関するスケーラビリティは明らかになっていない．そこで本稿では，strong scaling 問題においてプログラム分割による消費エネルギー最適化手法を適用し，ノード数と消費エネルギー，消費エネルギー最適化効率の関係について議論する．評価結果より，ノード数の増加に従って標準動作 P-State 実行時の消費エネルギーは増加する傾向があることが分かった．また，消費エネルギー最適化を適用した strong scaling 問題ではノード数の増加に従って消費エネルギー最適化効率が向上し，大規模並列システムにおいてもプログラム分割による電力最適化が有効であることが分かった．

Prediction of Optimized Energy Consumption of Power-aware Large-scale Cluster System

Hideaki Kimura,[†,††] Takayuki Imada[†] and Mitsuhisa Sato [†]

Recently, it has become important to improve the energy efficiency of high performance PC cluster systems. Profile-based energy optimization method, which defines program regions that have the almost same characteristics and selects the best P-State by analyzing profiles, can achieve energy efficient computing. However, it is difficult to evaluate the energy efficiency in a large-scale cluster system, therefore the energy scalability has not been discussed. This paper investigates both energy efficiency achieved by an energy optimization with program partitioning and scalability of energy efficiency. As a result, we found that the energy optimization method can reduce energy consumption in the large-scale PC cluster system. Increasing the number of nodes can achieve more effective energy optimization, though it wastes more energy consumption. The results indicated that the energy optimization with defining program regions is effective for the strong scaling problem.

1. はじめに

PC クラスタに代表される並列システムを構築する上で，システム全体の消費電力が重要な要素になっている．システムの消費電力増加によって電力コストが増加し，高価な冷却システムが必要となる．また，信頼性や実装密度の低下といった大規模並列システムを構築する上での阻害要因にもなる．このような問題に対し，動作 P-State（動作周波数と動作電圧の組）を動的に制御する DVFS（Dynanmic Voltage and Frequency Scaling）機構を制御することにより消費エネルギーを削減する手法が提案されている．プログラムの特性に応じて DVFS 機構を適切に制御することで，高い性能と省電力性を両立することができる[1),2)]．

また，PC クラスタなどの並列システムで頻繁に実行されるアプリケーションの多くはソースコード上の特定部を非常に多くの回数繰り返し実行する傾向があり，ソースコードに対して最適化のためのコード挿入が有効である．これにより，わずかなプログラム修正で最適化を実現でき，高い性能と大幅な消費エネルギー削減を同時に達成できる．

このような並列システム向け消費エネルギー削減手法として，プログラム分割とプロファイル取得による消費エネルギー最適化がある[3),4)]．プログラム分割とプロファイル取得による消費エネルギー最適化は，以下の手順によって実現できる．

- 特性に応じてプログラムを複数の領域に分割し，動作 P-State 制御のためのコードを挿入する
- プログラムを様々な動作 P-State で実行し，実行時間や消費電力に関するプロファイルを取得する
- 取得したプロファイル情報から定義した領域の最適動作 P-State を決定する

† 筑波大学大学院 システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†† 日本学術振興会 特別研究員
Research Fellow of the Japan Society for the Promotion of Science

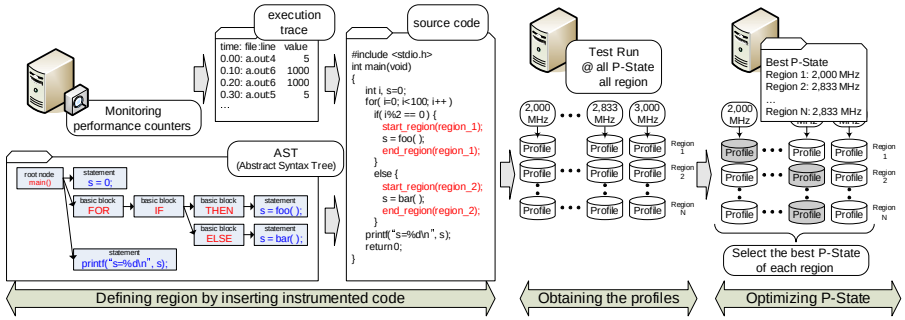

図 1 プログラム分割とプロファイル取得による消費エネルギー最適化

これまでに提案された電力最適化の多くは，“どのようにプロファイルを取得するか”，“どのように最適実行系列を決定するか”，“最適化指標は何か” に着目しており，対象となるシステムの最適実行系列を求めることが目的であった．プロファイルを取得するための予備実行と実際の最適化実行では同一の環境が必要であり，ノード数が異なる環境の最適化は考慮されていない．また，大規模システムにおいて詳細な電力プロファイルを取得する事は困難であり，実測による評価の多くは小規模システムを用いていた．このため，大規模システムにおけるプログラム分割による電力最適化の効果は明らかではない．ノード数と最適動作 P-State，消費エネルギー最適化効率の関係が明らかになれば，ノード数が異なる環境の消費エネルギー最適化効率を推測できる．また，これまで明らかでなかった大規模システムにおける消費エネルギー最適化の有効性を示すことができる．

そこで本稿では，プログラムの特性に応じて複数の領域を定義し動作 P-State を制御する消費エネルギー最適化とノード数の関係について述べる．具体的に，以下について議論する．

- ノード数と最適動作 P-State の関係を明らかにし，異なるノード数での最適化実行を支援する
- 消費エネルギー最適化効率をモデル化し，大規模システムにおける消費エネルギー最適化の有効性を明らかにする

本稿は，strong scaling 問題におけるプロファイルを用いた最適化時の消費エネルギー最適化効率を対象とし，ノード数と最適動作 P-State，消費エネルギー最適化効率の関係について議論する．また，プロファイルを用いた電力最適化を大規模システムに適用した時の消費エネルギーを予測し，消費エネルギー最適化に関するスケーラビリティを議論するとともに，大規模システムにおける消費エネルギー最適化の可能性を示す．

本稿の構成は以下のようになっている．次章では本稿で扱う消費エネルギー最適化手法の概要と課題を述べ，第 3 章で大規模システムの消費エネルギー予測手法について述べる．第 4 章で評価結果を示し，第 5 章で消費エネルギー最適化に関するスケーラビリティについて議論する．第 6 章で関連研究について述べ，最後にまとめと今後の課題を述べる．

2. プログラム分割による電力最適化

2.1 プログラム分割による電力最適化

プログラムを複数領域に分割し，各領域の特性に応じて適切な動作 P-State（動作電圧と動作周波数の組）を選択することで並列システム全体の消費エネルギーを削減することができる．一般的に，メモリアクセスや通信などの高い CPU 負荷を必要としない場面で低い動作 P-State を選択することで，わずかな性能低下と大幅な消費エネルギー削減を同時に達成できる．また，高性能計算向けアプリケーションの多くはソースコード上の特定部分を非常に多くの回数繰り返し実行する傾向があり，ソースコード中にコードを挿入しプログラムを複数領域に分割する最適化が有効であることが知られている．

本稿で対象とする “プログラム分割による電力最適化” は以下の手順によって実現可能である．

- プログラムの特徴に応じてソースコード上に最適化のためのコードを挿入し，プログラムを複数の領域に分割する
- すべての動作 P-State で予備実行し，定義したプログラム領域ごとに実行時間，消費電力に関するプロファイルを取得する
- 取得したプロファイルを解析し，定義した領域の最適動作 P-State を決定する

図 1 にプログラム分割による電力最適化の流れを示す．まず，動作 P-State を制御する単位である複数の領域にプログラムを分割する．我々はパフォーマンスカウンタのトレース情報とソースコードからコード挿入位置を決定し，プログラムを複数領域に分割する手法を提案している[5]．この手法を用いることで，以下の条件を同時に満たすコード挿入を実現する．

- ソースコード上で似た処理特性の部分を同一領域

と定義する
- 挿入したコードが非常に多くの回数呼ばれることを回避し，プログラム本来の特性を大幅に変化させない

次に，定義した領域の実行時間，消費エネルギーに関するプロファイルを取得し，最適実行系列を決定する．すべての動作 P-State でプログラムを実行することでプロファイルを取得し，プログラム全体の消費エネルギーや EDP（Energy Delay Product）が最小となる最適動作 P-State 系列を決定する．各領域の消費エネルギーが最小となる動作 P-State を選択するのみならず，動作 P-State 変更のためのオーバヘッドを考慮した手法[4] も提案されている．本稿では，最適化指標として消費エネルギーを用い，動作 P-State 変更のためのオーバヘッドを考慮した電力最適化手法[4] を用いる．

2.2 ノード数と電力最適化

これまでに，数多くのプログラム分割による電力最適化手法が提案されている．これらの手法の多くは最適実行系列の決定方法や最適化指標に着目しており，本実行と同一の環境でプロファイルを取得する必要があった．ノード数に関しても，プロファイル取得時と本実行時で同一でなければならず，ノード数の変更に伴いプロファイルの再取得が必要であった．また，大規模並列システムで詳細な電力測定を行う事は困難であり，小規模システムを用いた評価が行われるのみであった．

本稿では，データ分割による並列化，特に対象プログラムの問題サイズを固定しノード数を増加させる strong scaling 問題を対象とする．本稿では以下の条件が成り立つと仮定する．

- ノード数の変化によって各ノードのタスク量のみが変化し，タスクの特性は変化しない．
- プログラム分割により定義した領域では特定の処理のみが行われ，領域内で処理内容は変化しない．

ノード数と消費エネルギー最適化の関係について考える．1 ノード当たりの消費電力は動作 P-State や処理内容に依存し，定義した領域の最適動作 P-State は領域の処理内容によって決まる．すなわち，ノード数によって処理内容が変化しなければ，1 ノード当たりの消費電力は変化しないと考えられる．

一方，ノード数の増加によってシステム全体の消費電力，消費エネルギー，消費エネルギー最適化効率は変化する．ノード数の増加によって実行時間は減少し，システム全体の消費電力が増加すると予測される．完全並列化可能な問題ではノード数の増加と実行時間の短縮によって消費エネルギーは変化しないと考えられるが，逐次処理や並列化オーバヘッドが存在する場合にはシステム全体の消費エネルギーは増加する．

本稿では消費エネルギー最適化効率 EE_{opt}（Efficiency of Energy optimization）を式 (1) と定義する．$E^{n,f^{max}}$ は標準動作 P-State でプログラムを実行した時の消費エネルギー，$E^{n,f^{opt}}$ は各領域の動作 P-State を最適化した時の最適化消費エネルギーを表す．ノード数の変化により定義した領域の消費エネルギーも変化するため，消費エネルギー最適化効率もノード数に依存すると考えられる．

$$EE_{opt} = \frac{E^{n,f^{opt}}}{E^{n,f^{max}}} \tag{1}$$

3. 消費エネルギー最適化効率予測

実測により大規模システムの消費エネルギーを詳細に評価することは難しい．そこで本章では，消費エネルギー最適化効率をモデル化し，任意のノード数でプログラムを最適化実行した時の消費エネルギーを予測する手法について述べる．

3.1 P-State 制御時の消費エネルギー

最適化対象となるプログラムにおいて，以下の 4 種類のタスクを定義する．

$\omega^{s,ON}$: オンチップアクセスを伴う逐次処理タスク
$\omega^{s,OFF}$: オフチップアクセスを伴う逐次処理タスク
$\omega^{p,ON}$: オンチップアクセスを伴う並列処理タスク
$\omega^{p,OFF}$: オフチップアクセスを伴う並列処理タスク

また，ω^s，ω^p，ω^{ON}，ω^{OFF}，ω を以下のように定義する．

$$\begin{aligned}
\omega^s &= \omega^{s,ON} + \omega^{s,OFF} \\
\omega^p &= \omega^{p,ON} + \omega^{p,OFF} \\
\omega^{ON} &= \omega^{s,ON} + \omega^{p,ON} \\
\omega^{OFF} &= \omega^{s,OFF} + \omega^{p,OFF} \\
\omega &= \omega^{s,ON} + \omega^{s,OFF} + \omega^{p,ON} + \omega^{p,OFF}
\end{aligned}$$

また，n をノード数，f を動作周波数と定義する．

プログラム実行時間 $T^{n,f}$ は，ノード数 n，動作 P-State（動作周波数 f）によって変化する．まず，動作 P-State 制御による影響を考える．低い動作 P-State を選択することにより，オンチップアクセスタスクの実行時間は増加する．オンチップアクセスタスクにおいて動作周波数と実行時間は反比例の関係にあり，実行時間は動作周波数 f と標準周波数 f^{max} の比で表現できる．一方，オフチップアクセスタスクでは動作 P-State によって実行時間は変化しない．したがって，動作 P-State 制御とプログラム実行時間 $T^{1,f}$ の関係は式 (2) で与えられる．

$$T^{1,f} = \left\{ \frac{\omega^{ON}}{\omega} \frac{f^{max}}{f} + \frac{\omega^{OFF}}{\omega} \right\} T^{1,f^{max}} \tag{2}$$

一方，プログラム実行時間はノード数によっても変化する．Amdahl's Law より，ノード数 n とプログラム実行時間 $T^{n,f^{max}}$ の関係は式 (3) で与えられる．

$$T^{n,f^{max}} = \left\{ \frac{1}{n} \frac{\omega^p}{\omega} + \frac{\omega^s}{\omega} \right\} T^{1,f^{max}} \tag{3}$$

また，プログラムを並列実行する場合，通信などの並列化オーバヘッド $T^{overhead}$ が発生する．この並列

化オーバヘッドは，ノード数 n，動作 P-State，問題サイズなどに依存し，対象とするプログラムにより異なる．並列化オーバヘッドを考慮すると，ノード数による実行時間の変化は式 (4) で与えられる．

$$T^{n,f^{max}} = \left\{\frac{1}{n}\frac{\omega^p}{\omega} + \frac{\omega^s}{\omega}\right\}T^{1,f^{max}} + T^{overhead} \tag{4}$$

式 (2)，式 (4) は互いに独立である．したがって，プログラムの実行時間 $T^{n,f}$ とノード数 n，動作周波数 f の関係は式 (5) で与えられる．

$$\begin{aligned} T^{n,f} &= \left\{\frac{1}{n}\frac{\omega^p}{\omega} + \frac{\omega^s}{\omega}\right\}\left\{\frac{\omega^{ON}}{\omega}\frac{f^{max}}{f} + \frac{\omega^{OFF}}{\omega}\right\} \\ &\quad \times T^{1,f^{max}} + T^{overhead} \\ &= \left\{\frac{1}{n}\frac{\omega^{p,ON}}{\omega}\frac{f^{max}}{f} + \frac{1}{n}\frac{\omega^{p,OFF}}{\omega}\right. \\ &\quad \left. + \frac{\omega^{s,ON}}{\omega}\frac{f^{max}}{f} + \frac{\omega^{s,OFF}}{\omega}\right\}T^{1,f^{max}} \\ &\quad + T^{overhead} \end{aligned} \tag{5}$$

次に，消費エネルギーについて考える．ノード数が変化した場合であっても処理内容や処理特性が変化しなければ単一ノードの消費電力は変化しない．すなわち，ある単一の処理を複数ノードで実行する時 $P^{n,f} = n \times P^{1,f}$ が成立する．プログラムを動作周波数 f で実行した時の平均消費電力を $P^{1,f}$ とすれば，基準ノード実行時の消費エネルギー $E^{1,f}$ は式 (6)，n ノード実行時の消費エネルギー $E^{n,f}$ は式 (7) で与えられる．

$$E^{1,f} = \left\{\frac{\omega^{ON}}{\omega}\frac{f^{max}}{f} + \frac{\omega^{OFF}}{\omega}\right\}T^{1,f^{max}} \times P^{1,f} \tag{6}$$

$$\begin{aligned} E^{n,f} &= \left\{\frac{1}{n}\Big(\frac{\omega^{p,ON}}{\omega}\frac{f^{max}}{f} + \frac{\omega^{p,OFF}}{\omega}\Big)\right. \\ &\quad \left. + \frac{\omega^{s,ON}}{\omega}\frac{f^{max}}{f} + \frac{\omega^{s,OFF}}{\omega}\right\}T^{1,f^{max}} \\ &\quad \times P^{n,f} + n \times E^{overhead} \\ &= \left\{\frac{\omega^{p,ON}}{\omega}\frac{f^{max}}{f} + \frac{\omega^{p,OFF}}{\omega}\right. \\ &\quad \left. + n\Big(\frac{\omega^{s,ON}}{\omega}\frac{f^{max}}{f} + \frac{\omega^{s,OFF}}{\omega}\Big)\right\}T^{1,f^{max}} \\ &\quad \times P^{1,f} + n \times E^{overhead} \end{aligned} \tag{7}$$

ただし，各ノードにおける並列化のための消費エネルギー増加分を $E^{overhead}$ とする．

3.2 プログラム分割による電力最適化時の消費エネルギー

プログラムを処理内容に応じて R 個の領域に分割し，各領域の特性に応じて適切な動作 P-State を選択することで並列システム全体の消費エネルギーを削減する．定義した領域 r におけるタスク量を以下のように定義する．

$\omega_r^{s,ON}$: 領域 r におけるオンチップ逐次処理タスク
$\omega_r^{s,OFF}$: 領域 r におけるオフチップ逐次処理タスク
$\omega_r^{p,ON}$: 領域 r におけるオンチップ並列処理タスク
$\omega_r^{p,OFF}$: 領域 r におけるオフチップ並列処理タスク
$\omega_r = \omega_r^{s,ON} + \omega_r^{s,OFF} + \omega_r^{p,ON} + \omega_r^{p,OFF}$

この時，プログラム全体の最適化消費エネルギー $E^{n,f^{opt}}$ はプログラム分割によって定義した各領域における消費エネルギーの総和で表現できる．式 (8) に最適化時のプログラム全体の消費エネルギーを示す．

$$\begin{aligned} E^{n,f^{opt}} = \sum_{r=1}^{R}\Bigg[&\left\{\frac{\omega_r^{p,ON}}{\omega_r}\frac{f_r^{max}}{f_r} + \frac{\omega_r^{p,OFF}}{\omega_r}\right. \\ &\left. + n\Big(\frac{\omega_r^{s,ON}}{\omega_r}\frac{f_r^{max}}{f_r} + \frac{\omega_r^{s,OFF}}{\omega_r}\Big)\right\}T_r^{1,f^{max}} \\ &\times P_r^{1,f} + n \times E_r^{overhead}\Bigg] \end{aligned} \tag{8}$$

また，動作 P-State 制御を行わない時の消費エネルギーを式 (9) に示す．

$$\begin{aligned} E^{n,f^{max}} = \sum_{r=1}^{R}\Bigg[&\left\{\frac{\omega_r^{p}}{\omega_r} + n\Big(\frac{\omega_r^{s}}{\omega_r}\Big)\right\}T_r^{1,f^{max}} \\ &\times P_r^{1,f} + n \times E_r^{overhead}\Bigg] \end{aligned} \tag{9}$$

式 (8)，式 (9) において，$T_r^{1,f^{max}}$，$P_r^{1,f}$ は基準ノード数における領域 r の実行時間，消費電力である．また，$E_r^{overhead}$ は領域 r における 1 ノード当たりの消費エネルギーオーバヘッドである．これらの値は小規模システムにおいてプログラムを実行することにより取得可能である．電力測定が可能な範囲でプログラムを実行し，実測によって値を取得する．また，$\omega_r^{s,ON}/\omega_r$，$\omega_r^{s,OFF}/\omega_r$，$\omega_r^{p,ON}/\omega_r$，$\omega_r^{p,OFF}/\omega_r$，および $E_r^{overhead}$ を与えることにより，任意のノード数で最適化実行した時の消費エネルギー，消費エネルギー最適化効率を求めることができる．

$\omega_r^{s,ON}/\omega_r$，$\omega_r^{s,OFF}/\omega_r$，$\omega_r^{p,ON}/\omega_r$，$\omega_r^{p,OFF}/\omega_r$ は小規模システムで学習することによって求める．式 (2)，$\omega^{OFF}/\omega = 1 - \omega^{ON}/\omega$ より，式 (10) を得る．

$$\begin{aligned} T^{1,f} &= \left\{\Big(\frac{f^{max}}{f} - 1\Big)\frac{\omega^{ON}}{\omega} + 1\right\}T^{1,f^{max}} \\ \frac{T^{1,f}}{T^{1,f^{max}}} - 1 &= \Big(\frac{f^{max}}{f} - 1\Big)\frac{\omega^{ON}}{\omega} \end{aligned} \tag{10}$$

領域 r を様々な動作周波数 f_r^i で実行した時の実行時間 T_r^{1,f^i} を取得する．式 (11) を最小化する ω_r^{ON}/ω_r は式 (12) で与えられる．

$$\sum_i\left\{\Big(\frac{T_r^{1,f^i}}{T_r^{1,f^{max}}} - 1\Big) - \Big(\frac{f^{max}}{f_r^i} - 1\Big)\frac{\omega_r^{ON}}{\omega_r}\right\}^2 \tag{11}$$

$$\frac{\omega_r^{ON}}{\omega_r} = \frac{\sum_i(\frac{f^{max}}{f_r^i} - 1)(\frac{T_r^{1,f^i}}{T_r^{1,f^{max}}} - 1)}{\sum_i(\frac{f^{max}}{f_r^i} - 1)^2} \tag{12}$$

同様に小規模システムでノード数を変更させつつプ

ログラムを実行し，ノード数 j の実行時間 $T^{j,f^{max}}$ を取得する．これらは互いに独立であるため，式 (13)，式 (14)，式 (15)，式 (16) が成り立つ．

$$\frac{\omega_r^{s,ON}}{\omega_r} = \frac{\sum_i (\frac{f^{max}}{f_r^i} - 1)(\frac{T^{1,f_r^i}}{T^{1,f^{max}}} - 1)}{\sum_i (\frac{f^{max}}{f_r^i} - 1)^2} \times \frac{\sum_j (\frac{1}{j} - 1)(\frac{1}{j} - \frac{T^{j,f^{max}}}{T^{n,f^{max}}})}{\sum_j (\frac{1}{j} - 1)^2} \quad (13)$$

$$\frac{\omega_r^{s,OFF}}{\omega_r} = \frac{\sum_i (\frac{f^{max}}{f_r^i} - 1)(\frac{f^{max}}{f_r^i} - \frac{T^{1,f_r^i}}{T^{1,f^{max}}})}{\sum_i (\frac{f^{max}}{f_r^i} - 1)^2} \times \frac{\sum_j (\frac{1}{j} - 1)(\frac{1}{j} - \frac{T^{j,f^{max}}}{T^{n,f^{max}}})}{\sum_j (\frac{1}{j} - 1)^2} \quad (14)$$

$$\frac{\omega_r^{p,ON}}{\omega_r} = \frac{\sum_i (\frac{f^{max}}{f_r^i} - 1)(\frac{T^{1,f_r^i}}{T^{1,f^{max}}} - 1)}{\sum_i (\frac{f^{max}}{f_r^i} - 1)^2} \times \frac{\sum_j (\frac{1}{j} - 1)(\frac{T^{j,f^{max}}}{T^{n,f^{max}}} - 1)}{\sum_j (\frac{1}{j} - 1)^2} \quad (15)$$

$$\frac{\omega_r^{p,OFF}}{\omega_r} = \frac{\sum_i (\frac{f^{max}}{f_r^i} - 1)(\frac{f^{max}}{f_r^i} - \frac{T^{1,f_r^i}}{T^{1,f^{max}}})}{\sum_i (\frac{f^{max}}{f_r^i} - 1)^2} \times \frac{\sum_j (\frac{1}{j} - 1)(\frac{T^{j,f^{max}}}{T^{n,f^{max}}} - 1)}{\sum_j (\frac{1}{j} - 1)^2} \quad (16)$$

4. 評　　価

4.1 評価環境

評価環境として，Intel Core2Quad Q9650 (標準動作周波数: 3,000 MHz) を搭載した 16 ノードのクラスタシステムを用いた．**表 1** にノード構成を示す．Intel Core2Quad Q9650 は 6 段階の動作 P-State（3,000 MHz，2,833 MHz，2,667 MHz，2,500 MHz，2,333 MHz，2,000 MHz）を利用することが可能である．

電力評価環境として，ホール素子を用いた電力測定システム PowerWatch[4] を用い，10 msec 間隔で消費電力を測定した．各ノードは電力測定装置を介して UPS に接続し，ノード単位での電力測定を行った．よって，評価に用いる消費電力値は動作 P-State 制御で制御可能なプロセッサの消費電力のみならず，メモリや HDD などの消費電力，電源における AC-DC 変換損失を含む．また，計算ノードのみを対象とし，動作状態に依らず一定の電力を消費するネットワークスイッチは測定に含めない．

評価ベンチマークとして，NPB（NAS Parallel Benchmarks）-3.3 よりカーネルベンチマーク 5 種を用いた．問題サイズは CLASS=C とし，標準入力データセットを利用した．

表 1 測定対象クラスタシステム

CPU	Intel Core2Quad Q9650
Default Frequency	3,000 MHz
Memory	DDR2 SDRAM 8GB
HDD	HGST HDT721010SLA360
Network	Gigabit Ethernet
Kernel	Linux 2.6.28-perfctr
MPI	OpenMPI 1.3.2

4.2 ノード数と最適化時の消費エネルギー

プログラムを複数領域に分割し，プロファイルを用いた電力最適化を行う．すべての動作 P-State でプログラムを実行し，実行時間と消費電力に関するプロファイルを取得する．プロファイルを比較し，消費エネルギーが最小となる最適動作 P-State を各領域に対して決定する．2 ノード，4 ノード，8 ノード，16 ノードで定義した領域の最適動作 P-State をそれぞれ求める．

表 2 に定義した領域と選択した最適動作 P-State（動作周波数）を示す．関数単位やループ単位など，処理内容に応じて様々な粒度で領域を定義した．主に関数を一つの領域と定義したものについては関数名，それ以外は処理内容に応じた領域名を示す．ノード数により異なる最適動作周波数を選択した場合，括弧内にノード数を付記する．全ノード数で同一の動作周波数を選択した場合は記述を省略する．

プログラム分割による電力最適化では，処理特性に応じて領域を定義する．しかしながら，非常に短い時間間隔で動作 P-State を変更した場合，プログラムの性能が大幅に低下することが知られている[5]．CG の領域 conj_grad() 内では様々な処理時間の短い処理を行っているが，それぞれを領域と定義した場合，プロファイル取得や動作 P-State 制御によるオーバヘッドが非常に大きくなる．そこで，これらの小さな処理をまとめて一つの領域と定義している．

並列化効率の高いオンチップアクセスタスク $\omega^{p,ON}$ が主要な領域では，最高動作 P-State を選択する傾向があり，主にオフチップアクセスを行うタスク $\omega^{s,OFF}$，$\omega^{p,OFF}$ が主要な領域では低い動作 P-State を選択することで消費エネルギーを削減している．逐次オンチップアクセスタスク $\omega^{s,ON}$ が主要となる領域は存在しなかった．

表 2 より，ノード数が変化してもほぼ同じ最適動作 P-State を選択していることが確認できる．本稿で対象とするデータ分割による並列化アプリケーションでは，演算・メモリアクセスを行う領域においてノード数が変化しても処理内容は変化しないことが期待される．そのため，多くの領域でノード数に依らず一定の最適動作 P-State を選択した．しかしながら，主に集合通信を行う領域では，ノード数の変更により異なる動作 P-State を選択することがあった．例えば，IS の領域 MPI_Alltoallv() では 4 ノード実行時のみ 2,500 MHz

表 2 定義した領域と最適動作周波数

benckmark	region 1	region 2	region 3	region 4	region 5
FT	evolve() 2,000 MHz	cffts1() 3,000 MHz	transpose_x_yz() 2,333 MHz	cffts2(), cffts1() 3,000 MHz	
MG	rprj3() 2,333 MHz	compute on the coarsest grid 2,000 MHz (2),(4),(8) 2,333 MHz (16)	interp() 2,000 MHz	resid() 2,000 MHz	psinv() 2,333 MHz
CG	conj_grad() 2,000 MHz				
EP	main loop 3,000 MHz				
IS	sort into bucket 2,000 MHz	determine the num. of key 2,000 MHz	MPI_Alltoallv() 2,667 MHz (2),(8),(16) 2,500 MHz (4)		

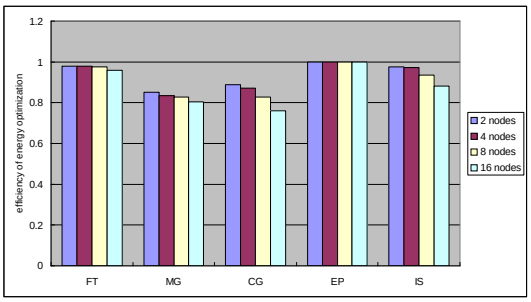

図 2 動作 P-State 最適化時の消費エネルギー

を選択した．2,500 MHz 実行時の消費エネルギーは 2,666 MHz 実行時と比較して 0.5 %大きかったが，これはプロファイル取得時の通信ミスによるものである．通信の衝突によって領域 MPI_Alltoallv() の消費エネルギーが僅かに逆転し，異なる最適動作 P-State を選択したと考えられる．MG の領域 compute on the coarsest grid においても，ノード数の変化によって異なる最適動作 P-State を選択した．この領域も主に集合通信を行っており，最適化のためのプロファイル取得時に実行時間，消費エネルギーのばらつきが大きかった．したがって，IS の領域 MPI_Alltoallv() と同様の理由によりノード数によって異なる最適動作 P-State を選択したと考えられる．

以上の結果より，主に演算やメモリアクセスを行う領域では，ノード数に依らず一定の動作 P-State を選択するといえる．また，実行時間が極端に短い領域，集合通信による実行時間のばらつきが発生する領域では，プロファイル取得結果によって異なる動作 P-State を選択する可能性がある．

図 2 に動作 P-State 最適化時の消費エネルギー最適化効率 EE_{opt} を示す．標準実行，最適化実行により取得した消費エネルギー実測値を式 (1) に適用し，各ノード数における消費エネルギー最適化効率を求める．

図 2 より，動作 P-State 最適化時に EP 以外のベンチマークにおいて消費エネルギーを削減していることが確認できる．EP では表 2 に示したように標準動作 P-State が最適であり，動作 P-State 制御によって消費エネルギーを削減できない．EP 以外のベンチマークでは，通信やメモリアクセスを行う領域において低い動作 P-State を選択することで消費エネルギーを削減した．動作 P-State 最適化によって，プログラム全体の消費エネルギーが増加することは無かった．

消費エネルギー最適化効率はノード数によって変化し，その傾向はベンチマークにより異なる．これは，定義した領域の実行割合がノード数によって変化するためである．EP ではひとつの領域のみ定義しており，消費エネルギー最適化効率はノード数に依らず一定である．同様に CG でもひとつの領域のみを定義し動作 P-State 最適化を行っているが，この領域内で様々な特性の処理を行っているため消費エネルギー最適化効率はノード数により変化した．MG，FT，IS では 2 ノードから 16 ノードにかけて消費エネルギー最適化効率が向上した．これは，ノード数の増加によって通信やメモリアクセスに要する消費エネルギー割合が増加し，低い動作 P-State 選択による消費エネルギー削減効果が大きくなったためである．

MG では定義したすべての領域で低い動作 P-State を選択し，消費エネルギーを削減した．定義した領域間の比較では，ノード数に依らず一定の割合で消費エネルギーを削減している．しかしながら，ノード数の変化によって各領域で消費するエネルギー割合が変化する．2 ノード実行時は領域 resid() で消費するエネルギーが最大であるが，ノード数の増加によって消費エネルギー最適化効率の高い領域 rprj3() や領域 psinv() で消費するエネルギーが増加した．このように MG ではノード数の増加により消費エネルギー最適化効率の高い領域が主導的となるため，プログラム全体の消費エネルギー最適化効率が向上した．FT や IS においても，ノード数の増加によって低い動作 P-State を選択可能な通信領域で消費するエネルギーの割合が増加し，プログラム全体の消費エネルギー最適化効率が向上した．

表 3 16 ノード最適化時の消費エネルギー

	実測値 [Wsec]	予測値 [Wsec]	誤差 [%]
FT	$1.321 * 10^5$	$1.385 * 10^5$	4.80
MG	$1.641 * 10^4$	$1.731 * 10^4$	5.48
CG	$1.901 * 10^5$	$1.683 * 10^5$	-11.5
EP	$1.619 * 10^4$	$1.630 * 10^4$	0.70
IS	$1.540 * 10^4$	$1.572 * 10^4$	2.10

4.3 消費エネルギー最適化効率予測

16 ノードで最適化実行した時の消費エネルギーを予測し，実測結果と比較する．提案モデルでは，小規模システムで定義した領域のタスク量を学習し，任意のノード数における最適化消費エネルギーを予測する．本稿では，2 ノード，4 ノード，8 ノードでタスク量を学習し，16 ノード実行時の最適化消費エネルギーを予測する．また，2 ノードを基準ノード数とし，実行時間，消費電力をそれぞれ実測によって取得する．

本稿で対象とするアプリケーションにおいて，並列化オーバヘッドの多くは集合通信である．strong scaling 問題では，ノード数の増加によって各ノードが扱うデータ量は減少する．一方，全ノードとデータ交換を行うためには最低でも $O(log_2 n)$ 回の通信が必要となる．そこで本稿では並列化による 1 ノード当たりの消費エネルギー増加分 $E^{overhead}$ を式 (17) と定義し，小規模システムにおいて α^f，β^f を学習した．

$$E^{overhead} = \frac{\alpha^f \log_2 n + \beta^f}{n} \quad (17)$$

消費エネルギー最適化時の消費エネルギー実測結果および，消費エネルギーシミュレーション結果を**表 3** に示す．本評価では，MG の領域 rprj3() と領域 compute on the coarsest grid，FT の領域 transpose_x_yz()，IS の領域 MPI_Alltoallv() をオーバヘッド領域としてシミュレーションを行った．また，実測結果では 16 ノード実行時に領域 compute on the coarsest grid で異なる動作 P-State を選択したが，8 ノード以下と同様に 2,000 MHz を用いるものとして消費エネルギーを予測した．

評価結果より，EP では 1%未満，IS では約 2%，FT，MG では約 5%の誤差で 16 ノード実行時の消費エネルギーを予測した．EP で唯一定義した領域 main loop は並列度の非常に高いオンチップアクセス処理を行う領域であり，$\omega^{p,ON}/\omega$ は 1.00 に，$\omega^{p,OFF}/\omega$，$\omega^{s,ON}/\omega$，$\omega^{s,OFF}/\omega$ は 0.00 に近い値となる．ノード数が変化しても集合通信などによるオーバヘッドは発生せず，高い精度で消費エネルギーを予測した．

IS の領域 MPI_Alltoallv() では，約 1%の誤差で消費エネルギーを予測した．この領域は集合通信を行っており，式 (17) によって高い精度で消費エネルギーを予測した．一方，他領域において誤差が大きく，プログラム全体としては約 2%の誤差となった．FT と MG では，定義したすべての領域で極端な予測ミスは見られなかった．CG では，評価した 5 ベンチマークの中でもっとも誤差が大きかった．これは，定義した単一領域内で様々な特性の処理を実行しているためである．処理内容に応じて詳細な領域設定を行うことで正確な消費エネルギー予測を行うことが可能であるが，動作 P-State 制御によるオーバヘッドを考慮すると実現は難しい．

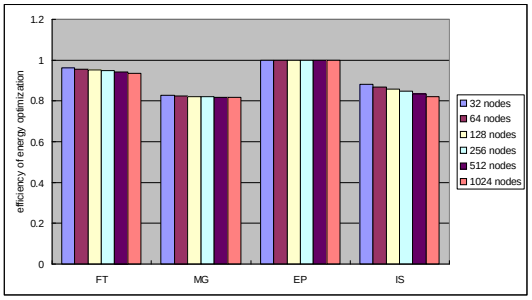

図 3 大規模並列環境での消費エネルギー最適化効率

5. 消費エネルギー最適化効率に関するスケーラビリティ

消費エネルギー，消費エネルギー最適化効率に関するスケーラビリティについて述べる．前章の評価より，演算やメモリアクセス領域においてノード数が変化しても最適動作 P-State は変化しないと考えられる．これは，ノード数の増加によって処理特性が変化しないためである．本章では，さらに大規模なノード数においても処理特性が変化しないと仮定し，評価を行う．また，CG では消費エネルギー予測に大きな誤差が含まれる可能性があることが分かっているため，本章では CG 以外のベンチマーク 4 種について評価する．

図 3 に消費エネルギー最適化効率のシミュレーション結果を示す．各ノード数において，標準動作 P-State でプログラムを実行した時の消費エネルギーを 1.00 とし，動作 P-State 最適化時の消費エネルギーを相対値で示す．また，**図 4** に FT，**図 5** に MG，**図 6** に EP，**図 7** に IS の消費エネルギーシミュレーション結果を示す．2 ノード標準動作 P-State でプログラムを実行した時の消費エネルギーを 1.00 とし，標準 P-State 時（std）と最適化時（opt）の消費エネルギーを領域別に示す．

図 3 より，EP 以外では動作 P-State 最適化により消費エネルギーが減少，EP では消費エネルギー最適化効率が変化しないという結果を得た．これは，前節で示した実測結果と同様である．複数領域を定義した FT，MG，IS ではノード数の増加に従って最適化による消費エネルギー削減量が増加している．また，消費エネルギー最適化効率も向上している．ノード数の増加に伴い，逐次オフチップアクセスタスク $\omega^{s,OFF}$，通信オーバヘッドによる消費エネルギーが増加する．

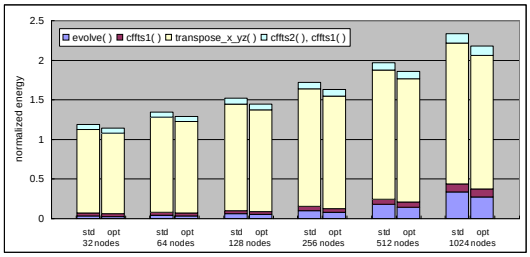

図 4 消費エネルギーに関するスケーラビリティ（FT）

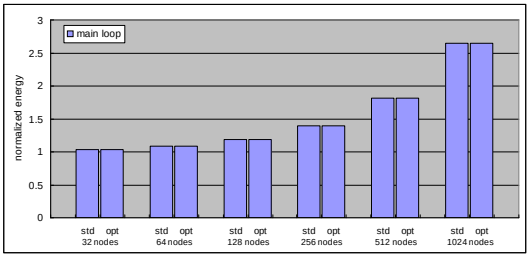

図 6 消費エネルギーに関するスケーラビリティ（EP）

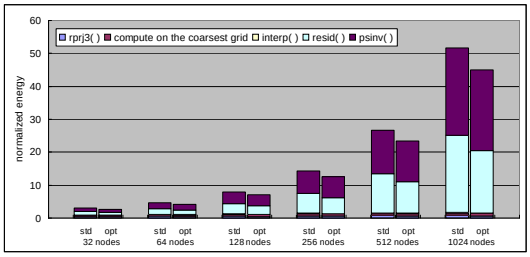

図 5 消費エネルギーに関するスケーラビリティ（MG）

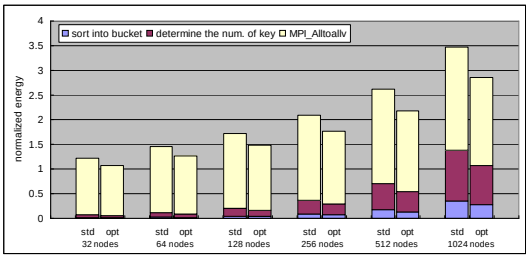

図 7 消費エネルギーに関するスケーラビリティ（IS）

これらの処理が主要な領域では低い動作 P-State を選択する傾向があり，ノード数の増加によって多くの消費エネルギーを削減できる．一方，高い動作 P-State を選択する傾向がある並列オンチップアクセスタスク $\omega^{p,ON}$ が主要な領域ではクラスタシステム全体の消費エネルギーはほぼ変化しない．この結果，複数領域の組み合わせからなるプログラムではノード数の増加により消費エネルギー最適化効率が向上する傾向がある．また，プログラム全体の消費エネルギー最適化効率は，最も並列度の低い領域の消費エネルギー最適化効率に近づく．

一方，プログラム全体の消費エネルギーはノード数とともに増加する．プログラムが完全に並列化可能であれば消費電力の増加と実行時間の短縮によって，システム全体の消費エネルギーは変化しない．しかしながら，逐次処理タスクや通信オーバヘッドによりクラスタ全体の消費エネルギーは増加し，クラスタシステム全体のエネルギー効率は低下する．

以上の結果より，strong scaling 問題において消費エネルギー最適化を行うことにより次のことがいえる．

- ノード数の増加に従って，消費エネルギー最適化の効果は大きくなる
- ノード数の増加に従って，プログラム実行に要する消費エネルギーは増加する

6. 関 連 研 究

並列システムにおいて動作 P-State，ノード数と消費エネルギーの関係を予測する手法が提案されている．Ge らは，プログラム全体を逐次/並列部，オンチップ/オフチップアクセス部に分割し，ノード数を変化させた際の消費エネルギーを予測する手法を提案している[6)7)]．Ding らは演算，メモリアクセス，通信に着目し FFT と密行列積実行時の消費エネルギーをシミュレートしている[8)]．アプリケーションの特性に応じて演算量・通信量を定義し，ノード数・動作 P-State と消費エネルギーの関係を予測している．これらの研究ではプログラム全体を一定の動作 P-State で実行することを前提としている．我々は，プログラム中の特性に応じて動作 P-State を最適化した時の消費エネルギー，およびその最適化効率に着目している．

Kamil らは，システム規模と消費エネルギーの関係を実機により評価している[9)]．Cray XT4 を用い，アプリケーション実行時におけるラック単位，全システムの消費電力特性を示している．しかしながら，消費エネルギー最適化には言及しておらず，動作 P-State 制御も行っていない．

Cho らは，並列化可能割合とプロセッサ数から動作 P-State を決定する手法を提案している[10)]．我々は，プログラムを複数領域に分割しプロファイルを取得することで最適動作 P-State を決定する手法を対象とし，消費エネルギー最適化効率を予測した．

ノード数と実行時間の関係として，Amdahl's Law が知られている[11)]．Barnes らは，回帰分析によりシステムのスケーラビリティを予測する手法を提案している[12)]．久保田らは，プログラムを演算部と通信部に分割し並列プログラムの性能予測を行っている[13)]．本研究は，プログラムを複数の領域に分割し，各領域で

回帰分析を行うことで並列化効率を求め，消費エネルギーに関するスケーラビリティを明らかにした．

7. おわりに

本稿では，プログラム分割による消費エネルギー最適化と実行ノード数の関係について述べた．高性能計算向けアプリケーションにおいて有効であると知られているプログラム分割による消費エネルギー最適化を様々なノード数で実現し，演算・メモリアクセスを行う領域において最適動作 P-State はノード数に依存しないことを確認した．また，小規模システムにおいて逐次/並列，オンチップ/オフチップアクセスタスクの割合を学習することで任意のノード数でプログラムを実行した時の消費エネルギーを予測する手法について述べた．さらに，標準動作 P-State 時と動作 P-State 最適化時の消費エネルギーから，消費エネルギー最適化効率とノード数の関係について考察した．シミュレーション結果より，大規模システムにおいてプログラム分割による電力最適化を適用することでプログラム実行に要する消費エネルギーは増加するが，消費エネルギー最適化効率が向上することが分かった．したがって，strong scaling 問題において消費エネルギー最適化は有効であるといえる．

今後の課題として，大規模な実クラスタ環境による評価が挙げられる．ノード数と消費エネルギー最適化効率の関係を大規模クラスタで検証し，検証結果をもとに消費エネルギー予測手法の高精度化を検討する．

謝辞 本研究の一部は，NEDO「グリーンネットワーク・システム技術研究開発プロジェクト（グリーン IT プロジェクト）/ エネルギー利用最適化データセンタ基盤技術の研究開発 / データセンタのモデル設計と総合評価」による．

参考文献

1) Chung-Hsing Hsu and Wu chun Feng. A Power-Aware Run-Time System for High-Performance Computing. In *SC05*, 2005.
2) Feng Pan, Vincent W. Freeh, and Daniel M. Smith. Exploring the Energy-Time Tradeoff in High-Performance Computing. In *IPDPS'05*, 2005.
3) Rong Ge, Xizhou Feng, and Kirk W. Cameron. Performance-constrained Distributed DVS Scheduling for Scienti c Applications on Power-aware Clusters. In *SC05*, 2005.
4) Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi. Profile-based Optimization of Power-Performance by using Dynamic Voltage Scaling on a PC cluster. In *IPDPS'06 Workshop on HPPAC*, 2006.
5) Hideaki Kimura, Takayuki Imada, and Mitsuhisa Sato. Runtime Energy Adaptation with Low-Impact Instrumented Code in a Power-Scalable Cluster System. In *CCGrid'10*, 2010.
6) Rong Ge and Kirk W. Cameron. Power-Aware Speedup. In *IPDPS'07*, 2007.
7) Rong Ge, Xizhou Feng, and Kirk W. Cameron. Modeling and Evaluating Energy-Performance Efficiency of Parallel Processing on Multicore Based Power Aware Systems. In *IPDPS'09*, 2009.
8) Yang Ding, Konrad Malkowski, Padma Raghavan, and Mahmut T. Kandemir. Towards Energy Efficient Scaling of Scientific Codes. In *IPDPS'08*, 2008.
9) Shoaib Kamil, John Shalf, and Erich Strohmaier. Power Efficiency in High Performance Computing. In *IPDPS'08*, 2008.
10) Sangyeun Cho and Rami Melhem. Corollaries to Amdahl's Law for Energy. *IEEE Computer Architecture Letters*, 2008.
11) G. M. Amdahl. Validity of the Single-Processor Approach to Achieving Large Scale Computing. In *AFIPS Spring Joint Computer Conference*, 1967.
12) Bradley J. Barnes, Barry Rountree, David K. Lowenthal, Jaxk Reeves, Bronis de Supinski, and Martin Schulz. A Regression-Based Approach to Scalability Prediction. In *ICS'08*, 2008.
13) 久保田和人, 板倉憲一, 佐藤三久, 朴泰祐. 大規模データ並列プログラムの性能予測手法と NPB 2.3 の性能評価. 情報処理学会論文誌, Vol. 40, No. 5, 1999.