電力制約下での性能向上を目的とした大規模並列計算機システム向け稼働計算ノード数決定法の提案

上原 有太[1]　稲富 雄一[2,3]　井上 弘士[2,3]

概要：近年，最大消費電力が制約を超過することを前提に大量のハードウェ アを設置し，アプリケーション特性に応じてシステム稼働時の実効消費電力が電力制約値を超えないよう制御する電力制約適応型システムの研究が注目を浴びている．本稿では，このような電力制約適応型システムの一例として，電力制約下において実効性能を高める計算ノード数決定法を提案する．32 個の計算ノードを有する計算機サーバシステムを用いた評価を行った結果，同一電力制約下において，従来実行方式と比べ最大で 1.77 倍の性能向上を実現した．

Parallelism Optimization to Boost Effective Performance of Power Constrained Supercomputers

Uehara Aruta[1]　Inadomi Yuichi[2,3]　Inoue Koji[2,3]

Abstract: This paper proposes a technique to improve parallel execution performance under power constraints. Power wall is one of the most serious issues for future exa-scale supercomputers. One of the solutions to tackle with this problem is to overprovision the hardware but manage the system power consumption dynamically. By means of assuming this kind of overprovisioned system, a methodology to find the best number of computing nodes to be used for application executions is proposed. The experimental results show that the proposed approach can achieve 1.77x performance improvement compared to a conventional execution scheme.

1. はじめに

スーパーコンピュータに代表される大型並列計算機システムの性能向上は，トランジスタ集積度の向上に伴う計算ノード性能の向上，ならびに，計算ノード数の増加によるシステム規模の拡大によって支えられてきた．しかしながら，2011 年に世界最高性能を達成した京コンピュータで約 13MW，2013 年現在での世界最速スパコンである Tianhe-2 では約 18MW という大きな電力を消費している [1]．米国 DARPA（Defense Advanced Research Projects Agency）の報告 [2] では現実的に供給可能な電力は 20MW とされており，今後のスパコン開発においては消費電力の増大が深刻な問題となっている．一方で，スーパーコンピュータ上で動作させるアプリケーション・プログラムのシステムへの要求は多様化しており，演算性能で 100 倍，メモリ帯域で 1000 倍，メモリ容量で 1000 倍もの差が要求仕様に生じるという報告がある [3]．したがって，次世代のエクサスケール・スーパーコンピュータを実現するためには，アプリケーション特性に基づく様々な要求仕様に対応できる柔軟性を有し，その上で，与えられた電力バジェットを効率的に利用してアプリケーションの実行性能を最大化するための技術開発が不可欠となる．

このような課題に対し，現在我々は，電力バジェットこ

[1] 九州大学 大学院システム情報科学府情報知能工学専攻
Dept. of Advanced Information Technology, Kyushu Univ.
[2] 九州大学 大学院システム情報科学研究院情報知能工学部門
Dept. of Advanced Information Technology, Kyushu Univ.
[3] 科学技術振興機構/CREST
JST/CREST

そが考慮すべき最重要資源であるとの考えに基づき，電力制約適応型システムに関する研究開発を進めている [4]．従来のスパコン設計法では，システム全体が稼働した際に消費される理論ピーク消費電力 (定格消費電力，いわゆる熱設計電力) が制約を越えない範囲でハードウェア資源を投入する．これに対し，電力制約適応型システムでは，最大消費電力が制約を超過することを前提に大量のハードウェアを設置する．そして，アプリケーション特性に応じてシステム稼働時の「実効消費電力 (実際に消費する電力)」が電力制約値を超えないよう制御する．このようなシステム実装方式を「オーバプロビジョニング方式」と呼ぶ．これまでに，科学技術計算向けアプリケーションの電力特性に関する解析等 [5] を行ってきた．しかしながら，オーバプロビジョニングに基づくシステム制御法とその効果に関する議論は未だ行われていない．

そこで本研究では，オーバプロビジョニングに基づく並列計算機システム構成を前提とし，電力制約下において実効性能を高める計算ノード数決定法を提案する．また，ベンチマーク・プログラムを用いた定量的評価を行い，その有効性を示す．一般に，並列計算機システムの性能は，計算ノード単体性能，ならびに，システムに搭載された総計算ノード数に大きく依存する．通常，システムが搭載可能な総計算ノード数は，設置可能面積（面積制約）と利用可能最大電力量（電力制約）によって決定されるが，本研究では電力制約にのみ着目する（すなわち，十分な設置面積を有する）．そして，各計算ノードでは消費電力量の設定（電力キャッピング）が可能であると仮定する．これにより，「各計算ノードへの電力キャッピングの度合い」と「電力制約下で稼働可能な計算ノード数」が変更可能となる．提案方式では，このようなオーバプロビジョニングを施した並列計算機システム構成を前提とし，アプリケーションが有する電力性能特性，並列性，ならびに，計算ノード未使用時の消費電力を考慮することで，電力制約下において性能が最大となる計算ノード数（ならびに各計算ノードで使用可能な消費電力バジェット）を決定する．本方式では，単一計算ノードを用いた 1 回，ならびに，複数計算ノードを用いた 3 回の事前実行を必要とする．これらの事前実行において実行時間と消費電力値を測定することにより最適な計算ノード数を求める．ベンチマークを用いた定量的評価を行った結果，多くの場合で最適な計算ノード数を決定でき，従来の並列計算機システム構成法と比較して最大で約 1.77 倍の性能向上を実現することができた．

本稿の構成は以下の通りである．まず，第 2 節で HPC（High Performance Computing）アプリケーションを対象とした電力性能特性の詳細を解析する．特に，計算ノードに割当て可能な電力バジェットを変化させた際の性能への影響，ならびに，オーバプロビジョニング方式による性能向上の可能性を定性的かつ定量的に解析する．次に第 3 節では，プロファイル情報に基づき計算ノード数を決定する方式を提案する．そして，第 4 節でベンチマーク・プログラムを用いた定量的評価を行い，提案方式の有効性を明らかにする．最後に第 5 節で本稿をまとめる．

2. 電力制約を考慮した HPC アプリケーションの並列実行性能解析

2.1 想定する並列計算機システム

本稿で想定する並列計算機システムである電力制約適応型システムについて説明する．第 1 節で述べたように，電力制約適応型システムではオーバプロビジョニングにより豊富なハードウェア資源を搭載する．そして，システムの稼働に利用可能な消費電力を越えない範囲内において，システムを構成する様々なハードウェア要素に対して適切に電力バジェットを配分する．各ハードウェア要素では，与えられた電力バジェットに基づき電力性能特性を決定するハードウェア・パラメータを決定する．ここで，各ハードウェア要素への電力バジェットの配分は，当該ハードウェア要素の電力キャップ値 (消費可能な電力の期待値) を決定することで実現する．例えば，計算ノードを構成する CPU においては，インテル社が提供する RAPL（Running Average Power Limit）[6] を用いることで電力キャッピングが可能となり，その値に応じて電源電圧と動作周波数の値が自動で調整される．RAPL では計算ノードに搭載された DRAM に対する電力キャッピングもサポートしており，CPU と DRAM への電力キャップ値を適切に決定することで高性能化を実現することが可能である [5].

計算ノードの消費電力の内訳を考えた場合，CPU とメモリ (DRAM) が支配的となる場合が多い [7]．そのため，これらのハードウェア要素は計算ノードにおける電力キャッピング対象となり得ることが望ましい．しかしながら，計算ノードに搭載されるメモリ容量は増加傾向にある（その結果，消費電力が増大する）ものの，以前として CPU が多くの電力を消費している．例えば，文献 [7] では，DRAM は CPU が消費する電力の 25〜50% 程度と報告している．また，現状では CPU と比較して，DRAM での動的電力制御 (DVFS：Dynamic Voltage and Frequency Scaling など) は一般的ではない．そこで本研究では，消費電力キャッピングの対象は CPU のみとする．

また稼働計算ノード数を制限している場合，各計算ノードはアプリケーションを実行している計算ノード（以下，稼働計算ノードと呼称）と実行を行っていないアイドル状態の計算ノード（以下，アイドル計算ノードと呼称）の 2 つに分類できる．アイドル計算ノードは，アプリケーションの実行は行っていないが，ある程度の電力を消費していると考えられる．そのため，本稿ではアイドル計算ノードは常にある一定の電力を消費していると仮定する．

2.2 電力バジェット分配方法

本節では，電力バジェットの各計算ノードへの分配方法について考える．計算ノードへの電力バジェットの分配法として，

- 計算ノード均等分配：計算ノード間で計算の進行度合いを共有する必要はないが，計算ノード間の負荷の不均一が生じた場合，最も実行が遅い計算ノードに性能が律速される
- 計算ノード不均等分配：計算ノード間の負荷の不均一を改善できるが，計算ノード間で計算の進行具合を共有しなければならない

の 2 通りが考えられる．電力バジェットを付近等に分配する場合，計算ノード間の計算の進行度合いを共有する機構が必要となるため，本稿では全ての計算ノードに割り当てる電力バジェットは均等であるとする．

ここで，システム全体で使用できる総電力バジェットを B_{total} とする．この時，稼働計算ノード数を N とした時の CPU に割り当て可能な電力バジェット $B_{CPU}(N)$ は式 (1) で表される．また，その時の CPU の電力キャップ値を式 (2) で表す．

$$B_{CPU}(N) = \frac{B_{total} - TDP_{node} * \alpha * (N_{max} - N)}{N} - P_{other} \quad (1)$$

$$CAP_{CPU}(N) = min(B_{CPU}(N), TDP_{CPU}) \quad (2)$$

各変数は以下の通りである．

- B_{total}：システム全体の電力バジェット [W]
- $B_{CPU}(N)$：稼働計算ノード数が N の時の，1 ノードの CPU の電力バジェット [W]
- $CAP_{CPU}(N)$：稼働計算ノード数が N の時の CPU の電力キャップ値 [W]
- N_{max}：システム全体の計算ノードの数
- P_{other}：CPU 以外のハードウェアの消費電力 [W]
- TDP_{node}：計算ノードの TDP[W]
- TDP_{CPU}：CPU の TDP[W]
- α：TDP_{node} に対するアイドル計算ノードの消費電力の比率

ただし，メモリやディスクといった CPU 以外のハードウェアは常に P_{other} だけ電力を消費しているとしている．

2.3 稼働計算ノード数と CPU 電力バジェットの関係

本節では，稼働計算ノード数が配分可能な CPU 電力バジェット量に与える影響を解析する．本解析では，九州大学情報基盤研究開発センターに設置されたスーパーコンピュータ PRIMEAGY CX400 を用いた．CX400 の諸元を表 1 に示す．

ここで，システム全体で消費可能な電力 Btotal を 8,400W とし，この電力制約を満たす範囲において稼働計算ノード数 N を 1 から 32 に変化させた際の $CAP_{CPU}(N)$ を図 1 に示す．横軸は稼働計算ノード数，縦軸は CPU の電力キャップ値 $CPU_{CAP}(N)$ を表す．本図では計算ノードのアイドル時（未稼働時）の消費電力を表すパラメータ α を 0〜0.5 まで変化させた場合を示している．この図より，稼働計算ノード数の増加に対して $CAP_{CPU}(N)$ が一定な区間，ならびに，単調減少する区間が存在することが分かる．前者は，計算ノードの TDP に対して十分な（TDP 以上の）電力バジェットを稼働計算ノードに割り当てる事が可能な範囲である．例えば，α が 0 の場合，20 台の計算ノードを TDP で稼働させることができる．その後，稼働計算ノード数の増加に伴い稼働計算ノード当たりに分配可能な電力バジェットが減少する．図 2 は図 1 における $CAP_{CPU}(N)$ の微分値（稼働計算ノード数を 1 台増加することにより生じる $CAP_{CPU}(N)$ の変化量）である．この図より，稼働計算ノードの増加により，各稼働計算ノードに対して電力キャッピングを施す必要が生じた瞬間（例えば，α が 0 の場合，稼働計算ノード数が 20 台から 21 台へと変化した時）に $CAP_{CPU}(N)$ が大幅に減少し，その後は緩やかに電力バジェットが減少することが分かる．これらの結果から，以下のことが明らかになった．

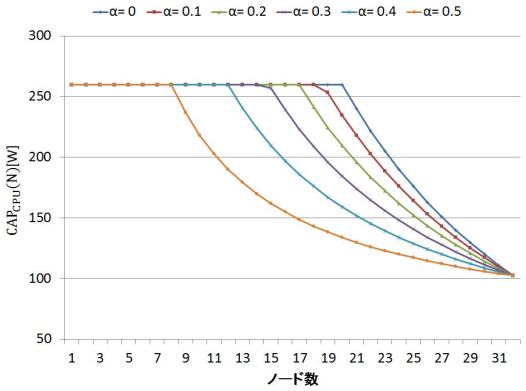

図 1 α に応じた $CAP_{CPU}(N)$

- ある電力制約下において稼働計算ノード数を増加させると，各稼働計算ノードに割り当てられる電力バジェットが計算ノードの TDP を下回ることにより CPU の電力キャッピングが必要となる．
- 各稼働計算ノードに施される CPU 電力キャッピングの量は，稼働計算ノード数の増加に減少する（CPU 電力キャッピングが有効になった瞬間が最も多くの電力バジェットを削減される）．
- 稼働計算ノード数を増やした場合，アイドル計算ノードが消費する電力の割合 α の増加に伴い，CPU の電力キャッピングを施す必要が生じる最小稼働計算ノード数，ならびに，稼働計算ノード当りの電力バジェット減少量は徐々に小さくなる．

表 1　CX400 の諸元

最大計算ノード数（N_{max}）	32	CX400 の仕様書より引用
CPU	Intel Xeon E5-2680	
1 ノードあたりの CPU 数	2	
1 ノードあたりのコア数	16（8 コア× 2）	
CPU の最大動作周波数	2.70[GHz]	
1 つの CPU の最低動作電力	51[W]	
1 つの CPU の TDP_{CPU}	130[W]	
計算ノードのメモリ構成	8GB × 16 枚（計 128GB）	
TDP_{node}	420[W]	
B_{total}	8,400[W]	$TDP_{node} * 20$ として算出
P_{other}	160[W]	$TDP_{node} - TDP_{CPU}$ として算出

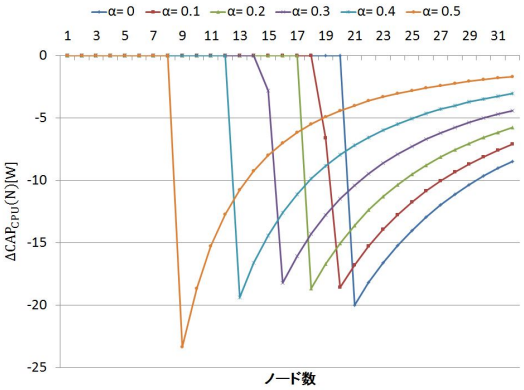

図 2　α に応じた $CAP_{CPU}(N)$ の微分値

2.4　定量的解析

2.4.1　実験概要と実験環境

本節では，実際の HPC アプリケーションを用いた定量的解析を行い，アプリケーションが有する電力性能特性，並列性，ならびに，アイドル計算ノードが消費する電力が実効性能に与える影響を明かにする．そして，これらの結果に基づき，オーバプロビジョニングによる性能向上効果を確認する．性能向上の比較対象として，1 ノードの消費電力を TDP_{node} として，電力制約を満たす最大の計算ノードを用いて実行した時の性能と比較する．

CPU の電力キャップ値は，式 (1)，(2) を用いて算出する．その後，α の値を変えることでアイドル計算ノードの消費電力を擬似的に変化させた後，同様の計測を繰り返す．それにより得られた性能（実行時間の逆数）を比較し，稼働計算ノード数とアイドル計算ノードの消費電力が性能に与える影響を確認する．CPU への消費電力キャッピングには RAPL を用いる．

2.4.2　ベンチマーク・プログラムと電力性能特性

本説では実験で使用した HPC ベンチマークの詳細を説明する．用いたベンチマークの内容は以下の通りである．

- BT：NPB（NAS Parallel Benchmark）[8] に含まれる，流体シミュレーションを行うベンチマーク・プログラムである．
- OpenFMO：九州大学，九州先端科学技術研究所で開発された FMO（Fragment Molecular Orbital method：フラグメント分子軌道法）プログラムである．FMO とは，タンパク質，DNA，糖鎖などの大規模生体分子に対する電子状態計算を行うために開発された並列処理向けの計算手法である．
- MHD シミュレーション：太陽風と呼ばれる磁場を伴った太陽から吹いてくるプラズマと惑星の磁場の相互作用によって形成される惑星磁気圏シミュレーションプログラムである [9]．

BT と MHD シミュレーションは MPI 並列化されているアプリケーションである．今回実験を行った環境では 1 ノードあたりのコア数が 16 であるため，1 ノードにつき 16MPI プロセスを立ち上げて実行を行った．ただし，MHD シミュレーションは問題サイズが MPI プロセス数に依存しているため，一律の問題サイズを維持できない計算ノード数に関しては実行を行っていない．一方，OpenFMO は MPI と OpenMP のハイブリット並列のアプリケーションである．そのため，1 ノードあたり 1 つの MPI プロセスを立ち上げ，各 MPI プロセスがスレッドを 16 個生成して並列実行を行うよう実行した．

まず，各ベンチマーク・プログラムの並列化効果について議論する．図 3 に各アプリケーションの並列性を示す．横軸は稼働計算ノード数，縦軸はアプリケーションごとに 1 ノードで実行した時の性能で正規化した正規化性能である．この結果より，BT と MHD シミュレーションはほぼ稼働計算ノード数に比例した性能向上を達成できており，並列性の高いアプリケーションであることが分かる．一方，OpenFMO は性能が頭打ちになっており，最大で 6 倍程度の性能向上しか達成できていない．そのため，並列性の低いアプリケーションであることが分かる．

また，図 4 に電力キャップ値に応じた 1 ノードでの実行性能を示す．横軸は電力キャップ値，縦軸は電力キャップ値が 51[W] の時の性能で正規化した正規化性能である．この図より，電力キャップ値がアプリケーションの実行に与える影響が分かる．各アプリケーションにおいて，ある電力キャップ値までは線形に性能向上しており，それ以上電

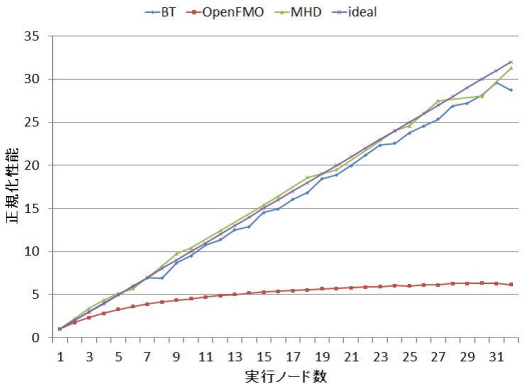

図 3 各アプリケーションの並列性

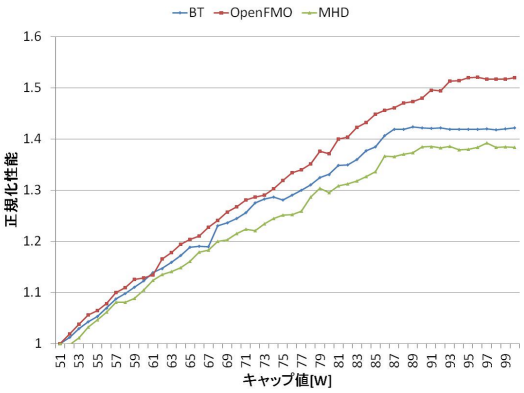

図 4 電力キャップ値に応じた 1 ノードでの実行性能

力キャップ値を大きくしても性能が横這いになっていることが確認できる．これは，性能が横這いになる電力キャップ値の時に CPU に対して十分に電力が供給されており，CPU が定格動作周波数で動作していることを示している．

2.4.3 結果

電力バジェットに応じた各アプリケーションの性能の推移を図 5〜7 に示す．図 5 は BT，図 6 は OpenFMO，図 7 は MHD シミュレーションのものである．各図において，横軸は稼働計算ノード数，縦軸は従来手法の性能で正規化した正規化性能である．凡例は，α=0〜0.4 は α がそれぞれの値の時，no CAP はキャッピングを行っていない時の実行結果を表している．

まず，電力制約適応型システムでの実行と従来手法との比較を行う．(B_{total}, α)=(3,360, 0.2),(5,040, 0.3) の時，最適な稼働計算ノード数で実行を行っても従来手法と比べて性能が低下している．これは，電力バジェットが少ないのに加えてアイドル計算ノードが消費している電力が大きく，稼働計算ノード数が従来手法未満となったためである．一方，それ以外の場合は電力制約適応型システムを用いることで従来手法と比較して性能が向上している．特に，並列性の高い BT，MHD シミュレーションでは，約 1.1〜1.8 倍の性能向上を達成しており，並列性の低い OpenFMO の場合でも最大で約 1.2 倍の性能向上を達成している．

次に，稼働計算ノード数と α が性能に与える影響を確認する．各 B_{total} において，動作周波数が低下し始める稼働計算ノード数から，性能が no CAP より低下している．この時，動作周波数が低下し始めた計算ノード数を N’，稼働計算ノード数が N の時の性能を Perf(N) とすると，N_a >N’ を満たす N_a に対して，アプリケーション，α によって Perf(N_a) と Perf(N’) の大小関係が異なっていることが分かる．

α の値が変化すると，図 2 に示す通り，$\Delta CPU_{CAP}(N_a)$ の値が変化する．また，図 4 に示す通り，計算ノード単体の性能は電力キャップ値に対して線形なので，その低下幅は $\Delta CPU_{CAP}(N_a)$ の絶対値に比例する．そのため，実行しているアプリケーションの並列度が高く，かつ α が大きい時に動作周波数低下による性能低下を計算ノード数増加による性能向上が上回ると，Perf(N_a)>Perf(N’) となる．一方，実行しているアプリケーションの並列度が低い，または α が小さい時，計算ノード数を増加させても動作周波数低下に見合うだけの性能向上を得られず，Perf(N’)>Perf(N_a) となる．

3. 提案手法

本章では，消費電力制約下における稼働計算ノード数の決定手法の提案を行う．また，提案手法から得られた計算ノード数（以下，提案稼働計算ノード数と呼称）と，全探索によって得られた最適な稼働計算ノード数（以下，最適稼働計算ノード数と呼称）と比較することで提案手法の評価を行う．

まず，動作周波数が低下し始めた点からの性能の変化は線形であると仮定する．つまり，動作周波数が低下し始めた稼働計算ノード数を N’ とすると，稼働計算ノード数が N’ より大きい時，性能は計算ノード数に対して単調増加，または単調減少であるとする．その仮定に基づき，以下の手順で提案稼働計算ノード数を決定する．

(1) 稼働計算ノード数を 1 として消費電力キャッピングを行わずに実行し，アプリケーション実行時の CPU の消費電力 $P_{CPU_{exe}}$ を計測する
(2) 式 (1)，(2) を用いて，電力キャップ値が $P_{CPU_{exe}}$ 以上となる最大の稼働計算ノード数 N_{nocap} と，電力キャップ値が CPU の最低動作電圧以上となる最大の稼働計算ノード数 N_{limit} を算出する
(3) 稼働計算ノード数を N_{nocap}，$N_{nocap}+1$，N_{limit} としてアプリケーションを実行し，その 3 つの中で最も性能が高い稼働計算ノード数を提案稼働計算ノード数とする

稼働計算ノード数が N の時の性能を Perf(N) とする．まず $1 \leq N \leq N_{nocap}$ の時，動作周波数は一定（定格動作周波数）である．そのため，稼働計算ノード数増加に伴い性能が向上するので，$N = N_{nocap}$ の時に性能が最大となる．

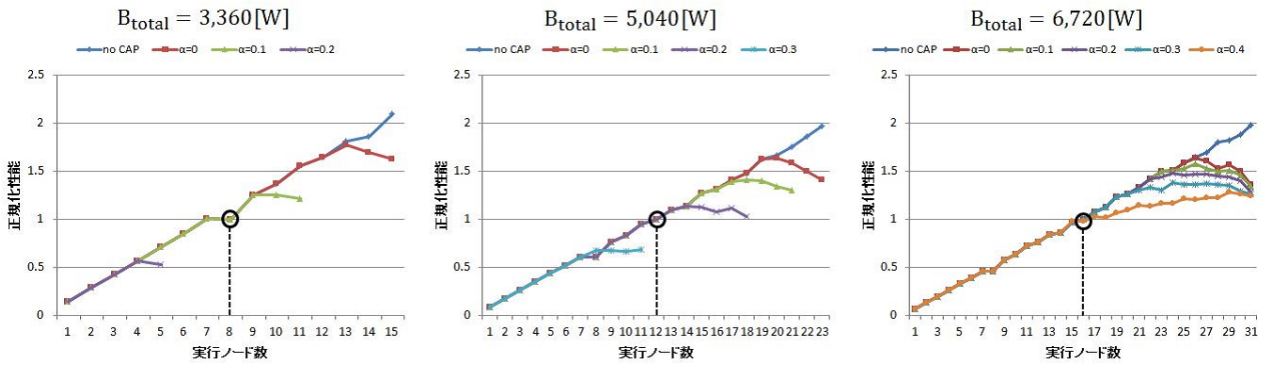

図 5 電力バジェット毎の BT の性能

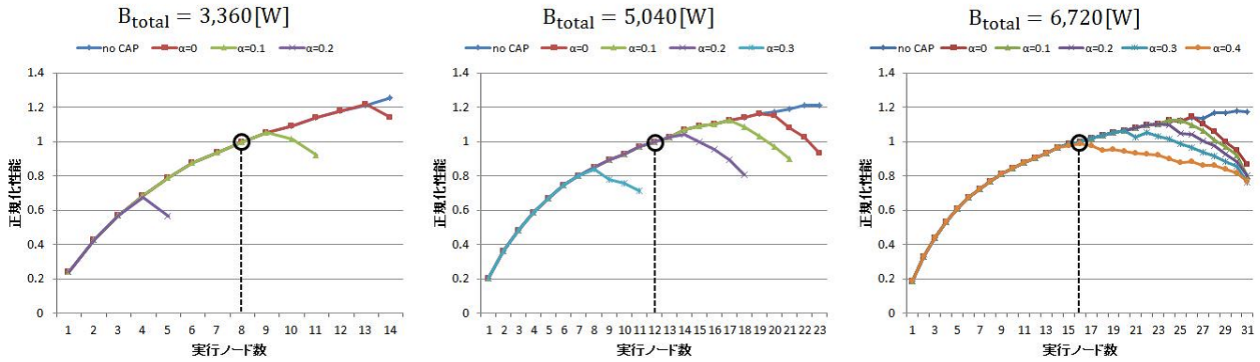

図 6 電力バジェット毎の OpenFMO の性能

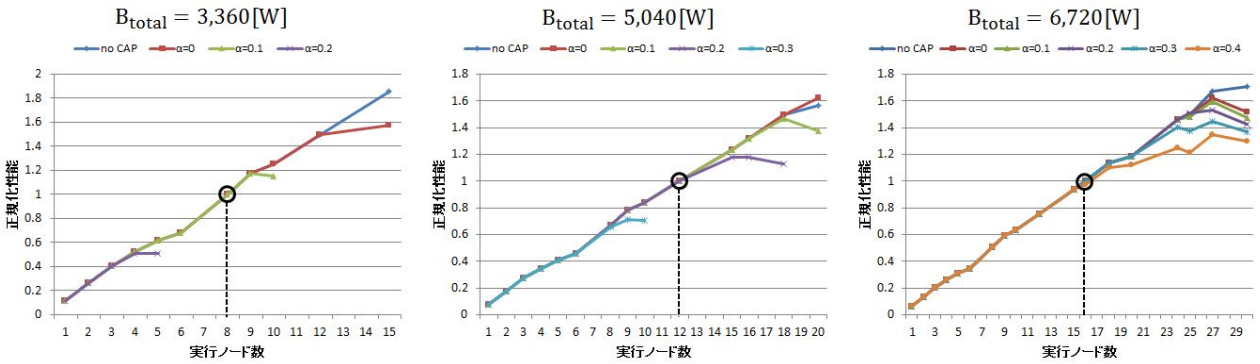

図 7 電力バジェット毎の MHD シミュレーションの性能

次に $N_{nocap} < N \leq N_{limit}$ の時を考える．仮定より，動作周波数が低下し始めた点からの性能の変化は線形であるため，$N_{nocap} < N \leq N_{limit}$ において性能が最大となるのは $N = N_{nocap} + 1$ の時か $N = N_{limit}$ の時のどちらかである．

以上より，性能が最大となりうるのは $N = N_{nocap}$，$N = N_{nocap} + 1$，$N = N_{limit}$ のいずれかのみであるため，これらの中で最も性能が高い稼働計算ノード数を提案稼働計算ノード数とする．

4. 評価

まず，提案稼働演算ノード数と，最適稼働演算ノード数の比較を行う．表 2〜4 に各アプリケーションにおける提案稼働演算ノード数と，最適稼働演算ノード数を示す．表 2 は BT，表 3 は OpenFMO，表 4 は MHD シミュレーションを示す．

B_{total}=3,360[W] の時，アプリケーション，α の値に関わらず，提案稼働演算ノード数と最適稼働演算ノード数が等しくなった．一方，B_{total}=5,040[W]，6,720[W] の時，提案稼働演算ノード数と最適稼働演算ノード数の間に違いが見られる．これは，提案手法では，CPU の動作周波数が低下し始めた点から計算ノード数増加に伴って性能が線形に変化すると仮定しているが，実際には線形ではなく，極大点が存在しているためである．

表 2 提案稼働演算ノード数と最適稼働演算ノード数の比較（BT）

$B_{total}[W]$	3,360			5,040				6,720				
α	0	0.1	0.2	0	0.1	0.2	0.3	0	0.1	0.2	0.3	0.4
提案稼働演算ノード数	13	9	4	20	18	14	11	27	26	24	22	31
最適稼働演算ノード数	13	9	4	20	18	14	11	26	26	24	24	29

表 3 提案稼働演算ノード数と最適稼働演算ノード数の比較（OpenFMO）

$B_{total}[W]$	3,360			5,040				6,720				
α	0	0.1	0.2	0	0.1	0.2	0.3	0	0.1	0.2	0.3	0.4
提案稼働演算ノード数	13	9	4	20	17	14	8	26	25	24	22	16
最適稼働演算ノード数	13	9	4	19	17	14	8	26	25	23	20	16

表 4 提案稼働演算ノード数と最適稼働演算ノード数の比較（MHD シミュレーション）

$B_{total}[W]$	3,360			5,040				6,720				
α	0	0.1	0.2	0	0.1	0.2	0.3	0	0.1	0.2	0.3	0.4
提案稼働演算ノード数	15	9	4	20	20	18	10	27	27	25	30	30
最適稼働演算ノード数	15	9	4	20	18	16	9	27	27	27	27	27

また，図 8〜10 に，稼働演算ノード数を提案稼働演算ノード数と最適稼働演算ノード数として各アプリケーションを実行した時の性能を示す．図 8 は BT，図 9 は OpenFMO，図 10 は MHD シミュレーションを示す．

横軸は電力バジェット [W] と α，縦軸は従来手法で実行時の性能で正規化した正規化性能である．図に示す通り，(B_{total}, α)=(3,360, 0.2),(5,040, 0.3) 以外の B_{total}，α の組の時，電力制約適応型システムで提案手法を用いることで従来手法と比べて性能が最大で約 77%向上している．また，提案稼働演算ノード数を用いた実行は，最適な実行と比較して最大でも 6%程度の性能低下に留まっている．

5. おわりに

次世代のスーパーコンピュータの開発に向けて，消費電力が最大の壁になると言われている．その解決策として電力制約適応型のの計算機システムに着目し，電力制約下において性能性能向上を目的とした稼働演算ノード数の決定法の提案を行った．提案手法を適用することで従来の実行方法と比較して最大で約 77%の性能向上を達成できることを示した．また，稼働演算ノード数が提案稼働演算ノード数の場合の実行と最適稼働演算ノード数の場合の実行を比較すると，提案手法は最適実行と比較して最大でも 6%程度の性能低下に留まることを示した．

謝辞 九州大学情報基盤研究開発センター深沢圭一郎氏をはじめとする「JST CREST：ポストペタスケールシステムのための電力マネージメントフレームワークの開発」研究メンバー諸氏に感謝します．

参考文献

[1] H. M. E. S. J. D. and H. S., “Top500 supercomputer sites. http://www.top500.org/.”
[2] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill *et al.*, “Exascale computing study: Technology challenges in achieving exascale systems,” 2008.
[3] N. e. a. Ishikawa Y. Maruyama, “Hpci 技術ロードマップ白書 (2012).”
[4] 科学技術振興機構, “ポストペタスケールシステムのための電力マネージメントフレームワークの開発.”
[5] 吉田匡兵, 佐々木広, 深沢圭一郎, 稲富雄一, 上田将嗣, 井上弘士, and 青柳睦, “Cpu と主記憶への電力バジェット配分を考慮した hpc アプリケーションの性能評価,” **情報処理学会研究報告**, vol. 2013, no. 21, pp. 1–8, 2013.
[6] Intel, “Intel:intel 64 and ia-32 architectures softeware developer’s manual (2012).”
[7] K. Yoshii, K. Iskra, R. Gupta, P. Beckman, V. Vishwanath, C. Yu, and S. Coghlan, “Evaluating power-monitoring capabilities on ibm blue gene/p and blue gene/q,” in *Cluster Computing (CLUSTER), 2012 IEEE International Conference on.* IEEE, 2012, pp. 36–44.
[8] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber *et al.*, “The nas parallel benchmarks summary and preliminary results,” in *Supercomputing, 1991. Supercomputing’91. Proceedings of the 1991 ACM/IEEE Conference on.* IEEE, 1991, pp. 158–165.
[9] K. Fukazawa, T. Ogino, and R. J. Walker, “Configuration and dynamics of the jovian magnetosphere,” *Journal of Geophysical Research: Space Physics (1978–2012)*, vol. 111, no. A10, 2006.

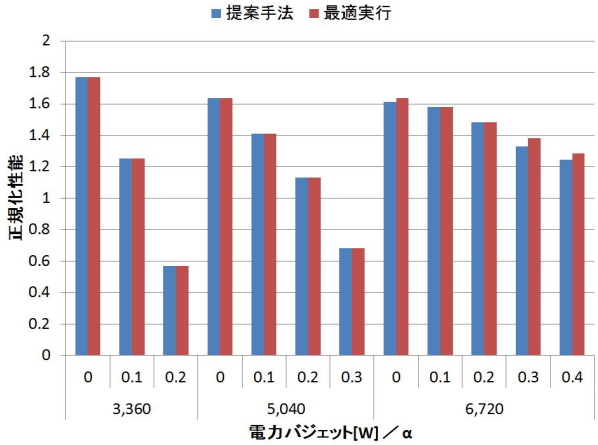

図 8 提案手法と最適実行の比較（BT）

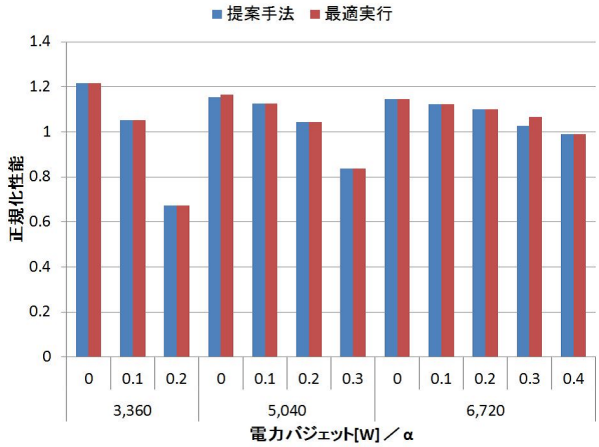

図 9 提案手法と最適実行の比較（OpenFMO）

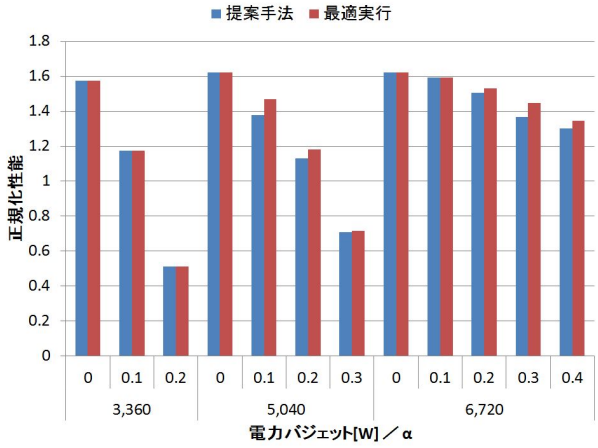

図 10 提案手法と最適実行の比較（MHD シミュレーション）