Vol.2017-HPC-160 No.1
2017/7/26

オーバープロビジョニング環境での大規模HPCシステムの電力と性能評価

坂本 龍一[†1,†3,a)] カオ タン[†1,†3] 近藤 正章[†1,†3] 井上 弘士[†2,†3] 上田 将嗣[†2] Tapasya Patki[†4]
Daniel Ellsworth[†4] Barry Rountree[†4] Martin Schulz[†4]

概要：HPC システムにおいて，最大消費電力が制約を上回ることを前提として大量のハードウェアを設置し，実行時に消費電力が制約を超えないように制御しつつジョブを実行するオーバープロビジョニング環境が注目されている．オーバープロビジョニング環境では電力制約内でイントールするノード台数が性能や電力特性，コストの面で重要なパラメタである．そこで本稿では，オーバープロビジョニング向けの電力制約を考慮したリソースマネージャの設計を示し，オーバープロビジョニング構成を想定した大規模なHPC システム上で，提案するリソースマネージャを用いて電力と性能の評価を行う．さらに，追加ノード台数とワークロードの特性に応じた最適なオーバープロビジョニング構成についての検討を行う．

1. はじめに

将来の HPC システムでは，消費電力がシステム設計や実行性能を制約する最大の要因となると考えられている．本原稿執筆時点で世界最高性能を誇る Sunway TaihuLight は 93 ペタフロップスの性能を達成するために 15MW 近い消費電力を要する [1]．しかし，現在の大規模計算機センターの電力設備状況や物理的な制約からするに，将来的にこれ以上の電力供給能力を持つ計算機センターを設置することは難しい．つまり，2020 年ごろに実現されるエクサスケール級のシステムでは，現在と同規模の 10〜20MW 程度の電力で，現在の 10 倍近い性能を達成することが必要である．

さらに，供給電力や熱設計消費電力制約の中でハードウェア資源量を決定し，システムのピーク消費電力が制約を超えないことを保証する従来の設計思想では，さまざまな特性を持つジョブを今後の大規模システムに対してスケールさせることは難しいと考えられる．このような背景のもと，我々はシステムのピーク消費電力が制約を超過することを積極的に許容し，ハードウェアが持つ電力ノブや計算に用いるハードウェア資源量を調整することで，限られた電力資源を計算・記憶・通信等の各要素に適応的に分配し，実効電力を制約以下に制御しつつ高い実行効率を得るハードウェアオーバープロビジョニング HPC システムの研究を進めている．

このようなオーバープロビジョニングシステムでは，ジョブの特性やシステムの運用状況等に合わせた電力管理・計算ノード資源管理が重要な役割となる．そのためこれらの計算ノードや電力資源を有効に利用するための研究が活発に行われている．一方で実際の大規模 HPC システムを想定したアルゴリズム実装や運用には大きなコストがかかるため，これらの研究の多くは小規模なシステムを対象としていたり，シミュレーションによる評価が行われている．そのため，大規模なシステムや実際のシステム運用に対する考慮が十分になされていないことが多い．さらに，電力ノブの制御には多くの場合，ルート権限が必要であるが実際の HPC 環境においてルート権限を得ることは運用上難しい．このように，(1) 実環境に対してどのように簡潔かつ安全に電力制御を行うかという課題がある．また，(2) これらの研究の多くは特定の HPC システムやシミュレーション環境に依存しているため性能の比較が難しいことも課題である．さらに，(3) オーバープロビジョニング環境においてどれだけのノードを追加するべきかという本質的な問題がある．

そこで，本研究ではこれらの課題に対しての解決策を与えることを目的に以下の貢献をする。(a) 様々な HPC システムに対応可能かつ，拡張可能なリソースマネージャを提案する．本リソースマネージャは HPC システムにおいて一般に利用されている SLURM スケジューラを拡張する．さらに，これらの拡張した SLURM を用いて，(b) 大規模

†1 東京大学
†2 九州大学
†3 科学技術振興機構 CREST
†4 Lawrence Livermore National Laboratory
a) r-sakamoto@hal.ipc.i.u-tokyo.ac.jp

な HPC システムを安定的に制御可能であることを示す．実際に九州大学の HA8000 システムを用いて，オーバープロビジョニング環境を想定した 965 ノードを安定的に制御できることを示す．さらに，本スケジューラを用いて様々な評価を行うことによって，(c) 追加するノード台数とシステム性能との関連を明らかにする．

2. 関連研究

本章では電力を考慮した資源管理手法の関連研究について示す．主に，ジョブスケジューリング，動的な電力制御，プロセスバラつき，性能予測手法に関する研究を示す．さらに，既存の SLURM が提供する機能についてまとめる．

2.1 ジョブスケジューリング

以前より，バックフィリング [2] 等のジョブスケジューリングを用いた資源利用の最適化に関する研究が盛んに行われている．オーバープロビジョニング環境が提案される以前より電力やエネルギーを考慮した研究 [3], [4], [5], [6], [7] が行われている．オーバープロビジョニング環境を想定した研究においてはノード資源と電力資源の双方を最適化する必要があり，電力制約値やコアの数等の最適化が重要である．Gholkar [8] や Patki ら [9] は moldable なジョブを考慮したスケジューリングを提案し，Patki は電力を考慮したバックフィリングを用いている．Sarood らは [10]moldable や malleable なジョブに割り当てるコア数を線形計画法を用いて最適化する手法を提案している．これらの研究では最適なジョブの配置を探索することで性能の向上を目指している．

2.2 動的な電源制御

前節で示したジョブスケジューリング手法はジョブの実行の直前に電力の割当てを静的に決めるものである．これは，動的に消費電力が変化する場合において電力資源を有効に利用することが難しい．そこで，動的に電力資源を管理する研究 [11], [12], [13], [14] が行われている．Cao らはプライオリティの低いジョブから高いジョブに対して電力を動的に再分配する手法を提案している．また，文献 [12] では周期的に電力をモニタリングし，電力を均等に分配する手法を提案している．

2.3 プロセスバラつき

近年，プロセス技術の微細化に伴い，同一型番のプロセッサや DRAM であっても電力特性に違いが生じることが問題 [15], [16] となっている．これらは電力キャッピングを行うオーバープロビジョニングシステムにおいて大きな課題となる．例えば，同一の電力キャップを多数のプロセッサに与えた場合，それぞれのプロセッサはプロセスバラつきにより電力特性が異なるため，それぞれのプロセッサの動作周波数が異なる結果となる．そのため全域で同期を取るようなジョブの場合，最も遅いノードに性能が律速されジョブの性能が大きく低下する．

これらの問題に対し 2 つのアプローチがとられている．1 つ目のアプローチはマルチコア CPU を想定し，電力制約の中でジョブが利用するコア数，物理コアの場所を最適化するアプローチ [17] である．一方で，稲富らや Cao らは多数のノード間でバラつきを考慮する手法 [18][13] を提案している．ジョブが利用するノード割当てを最適化することでアプリケーションの性能を向上させる手法を提案している．また，Gholkar らは 2 階層に渡るスケジューリング手法 [8] を提案している．マクロなレベルではシステム全体のジョブ割り当てを最適化し，ミクロなレベルではプロセスバラつきを考慮したスレッドスケジューリングを各ノード内で行っている．

2.4 性能予測

ジョブのスケジューリングや動的な電力制御を行うためには将来のジョブ実行を予測しつつジョブや電力の配置を決定することが重要である．そのため，電力キャップを与えた際のジョブの性能見積が重要になる．システムの電力制約を超えないためにも高精度な性能予測が重要といえる．Patki [9] はプロファイルを用いた回帰解析を用いることでジョブの実行時間の予測を行っている．Inadomi らは [18] はプロセスバラつきを考慮したジョブの性能を 2 回の試行により予測する手法を提案している．システムの立ち上げ時にすべてのプロセッサのバラつきを調査する．さらに，ジョブの実行直前に小規模なノードを用いてジョブの実行を行う．この際，電力キャップを与えずに実行した結果と，厳しい電力キャップを与えた場合の 2 通りの試行を行う．最後に，これらの試行の結果と立ち上げ時に取得したバラつきの情報から最適な電力キャップ値を計算する．また，Sarood らは Downey らが提案するスケーリングのモデル [19] を本に電力を考慮した強スケーリングの性能予測モデルを提案 [10] している．

2.5 SLURM 拡張

前節の多くの研究はシミュレータによる評価を行っていたり，簡易的なスケジューラを用いた評価がなされており実際の HPC システム環境を十分に想定しているとはいえない．実際の HPC システムではアカウント管理，プライオリティ等の多くの実用的な機能を提供しており，提案されているアルゴリズムを実環境に適応することは容易ではない．

いくつかの研究では実際に SLURM に対し拡張を行い，実際のシステムを用いて電力評価に取り組んでいる．Georgiou らは [20] 電力モニタを SLURM に組み込み HPC システムの消費電力計測を行っている．Bodas らは [21]3 つの

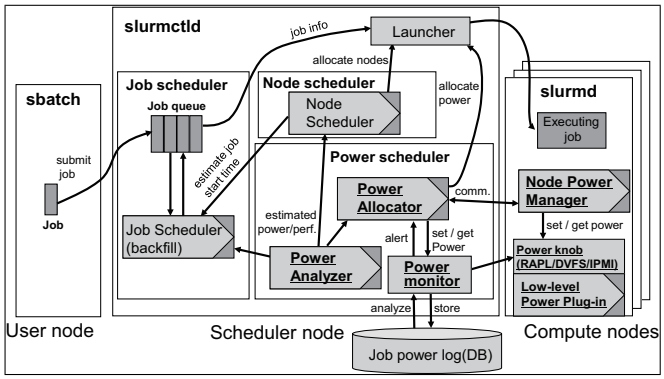

図 1 電力制約適応型 SLURM リソースマネージャの概要

異なる周波数制御アルゴリズムを提案し，SLURM に組み込み評価を行った．また，Cray のシステム向けに動的な電力制御を行う研究 [22] もなされている．さらに， [7], [23] らは小規模なクラスタ環境を用いた評価を示している．しかしながら，これらの研究は特定の HPC サイトに依存して実装されており，拡張性に乏しい課題がある．

そこで，我々は既存の SLURM を拡張し実用的な機能を提供し，かつ拡張化可能な電力制約適応型リソースマネージャを新たに提案する．提案する SLURM はジョブスケジューリングやノードスケジューリング，性能予測の機能をプラグインインタフェースとして提供することで，性能の予測や電力を考慮したジョブスケジューリング，動的な電力制御をサポートする．

3. 電力制約適応型リソースマネージャ設計

本章では我々が提案する電力制約適応型リソースマネージャの設計について述べる．本電力制約適応型リソースマネージャは様々な電力制御アルゴリズムの実装・評価を簡易化するための機能を提供する．多くの研究で用いられている電力制御アルゴリズムは，主に次に述べる 2 つの大きな要素から成り立っている．

- ジョブの消費電力の予測と実行時間の予測
- 計算ノード資源と電力資源の管理

図 1 は提案する電力制約適応型リソースマネージャの概要を示している．本システムは SLURM を基に電力制御のためのインタフェースを拡張する形で開発を進めている．SLURM はジョブスケジューリングアルゴリズムやノードスケジューリングアルゴリズムをプラグインとして実装できるようになっている．そのため，新しいジョブスケジューリングアルゴリズム等を開発する場合は，これらのプラグインの枠組みを用いることでアルゴリズムの主要な部分の実装に集中することができる．実際に SLURM のバックフィルジョブスケジューラは SLURM が提供するジョブスケジューリングプラグインインタフェースを用いて実装されている．電力制御の機能追加にあたり，我々はこれらのジョブスケジューリングプラグインインタフェース，ノードスケジューリングプラグインインタフェースを再利用し，さらに静的な電力制御と動的な電力制御を行うための新たなプラグインインタフェースを追加した．さらに，アルゴリズムの実装を行う研究者が容易に電力モニタリング，電力キャッピングを行えるように電力モニタリング機能と電力制御機能の追加も行った．

図 1 中のハイライトされている部分は新たに拡張を行った部分を示している．SLURM は主に 3 つのプログラムから構成されている．1 つ目はユーザーサイドのジョブサブミッション用プログラム (sbatch) である．さらに，これらの要求を受けシステム全体の資源管理を行うスケジューラ (slurmctld)，計算ノード上で動作するデーモン (slurmd) から構成される．ユーザーはジョブ実行のリクエストを SLURM に対し要求する．ジョブ実行の要求を受けたスケジューラ slurmctld は，要求されたジョブを一旦，ジョブキューに投入する．さらに，ジョブが要求するノード数，実行時間，プライオリティなどの情報を参考としジョブキュー内部のジョブの実行順序の最適化を行う．node scheduler はジョブが要求するコア数，メモリ容量，アクセラレータの有無などの情報を基にジョブに対する最適な物理的ノードの割当てを決定する．

我々は，power scheduler，node power manager を新たに追加した．さらに，既存の SLURM と連携するための low-level power plugin interface を追加した．power scheduler は分散されたノードの電力をモニタリングし電力の分配を行うことで，システム全体の電力資源を管理する役割を持つ．power scheduler は Power monitor，Power analyzer，Power allocator の 3 つから構成される．Power allocator と Power analyzer はプラグイン形式となっており，研究者は実装評価を行いたいアルゴリズムをこれらのプラグインインタフェースを用いて実装することができる．

3.1 電力制御のための基本機能

Power Monitor

Power monitor は周期的にノードの電力をモニタリングし，システム全体の消費電力を計算する役割を持つ．合わせて，ノードの電力のキャッピングの制御を行う．オーバープロビジョニング環境においてはソフトウェアのバグやノードの異常などが原因となり，システムの消費電力がシステムの電力制約を超える恐れがある．そのため，パワーモニタはこれらの異常事態をモニタリングする役割も担う．異常時はノードの電力キャップ値を小さくしたり，実行中のジョブを止めたりすることで対応する．

Power Analyzer

Power Analyzer はジョブの実行時間を予測する役割を持つ．利用可能なノード数や電力キャッピング等の制約情報からジョブの実行時間を予測する．ジョブスケジューリングプラグインは Power Analyzer プラグインが予測した

表 1　HA8000 システムの詳細

Total nodes	965
Sockets per node	2
Cores per socket	12
Total cores	23,160
Memory per node	128GB
CPU	Xeon E5-2670 v2 (2.5GHz)
Interconnect	Infiniband FDR
Operating System	Linux with kernel 2.6.32
MPI	Open MPI v1.7.3
Theoretical Peak (Rpeak)	1000 TFlop/s
Linpack Performance (Rmax)	712.5 TFlop/s

ジョブの実行時間情報を元にジョブの開始時刻を決定することで、システム全体のジョブスループットを最適化する．ノードスケジューリングプラグインもジョブスケジューリングプラグインと同様に将来のノードの利用状況を Power Analyser を用いて予測する．Power Analyzer はプラグインとして提供されており，様々な予測アルゴリズムを実装することができるようになっている．

Power Allocator

Power Allocator は Power Analyzer と協力し，静的な電力制御，動的な電力制御を行う機能を担う．Power Allocator はジョブの実行の直前に最適な電力割当てを検討し、また，システム全体の消費電力を最適化するために周期的にシステム全体の電力分配を行う機能を有する．また，ジョブの制約情報を本に将来の資源利用を予測する機能も有する。これらは電力を考慮したバックフィリングなどのような将来用いられる可能性のあるスケジューリングを考慮したものである．

Node Power Manager

Node Power Manager はノードサイドでの電力制御を行う役割を担う．スケジューラと協力しつつノード側で電力制御を行うことで，より細粒度に電源制御を行うことができるようになる．

Power Knobs と Low-level Power Plugins

様々なプロセッサがそれぞれ異なる電力計測の仕組みや電力キャッピングのための機能を提供している．例えば，DVFS, RAPL(Running Average Power Limit) [24], [25], IPMI[26] や PowerInsight [24], [27] 等がこれにあたる．それぞれ，電力制御のインタフェースが異なり，アルゴリズムの作成者はプロセッサごとにプログラムを作成する必要がある．これを解決するために Low-level power plugin はプロセッサに依存する低レベルな電源制御の機能を隠蔽する役割を持つ．それぞれのベンダのプロセッサに対応したプラグインを用いることで共通のインタフェースから電源を制御できるようにする。

4. 実装と開発環境

評価では提案する SLURM を大規模システム環境に導入し，提案する SLURM が大規模システムで実用的に電力制御が可能であることを確認する．さらに，提案する SLURM を用いて消費電力等を分析することで，オーバープロビジョニング環境の特性を明らかにする．本章では提案する SLURM をインストールする HPC システム構成，電源の管理ポリシーについて示す．ここで示す電源ポリシーは実際に提案する SLURM のプラグインインフェースを用いて実装し大規模環境での電力制御に用いる．

4.1 評価環境

本評価では九州大学の HA8000 を用いる．本システムは 965 台の計算ノードから構成され，それぞれの計算ノードに 2 ソケットの Xeon プロセッサと 128GB の DDR3 メモリが搭載されている．これらの詳細を表 1 に示す．利用する HA8000 の Xeon プロセッサは RAPL をサポートしており，CPU と DRAM の最大消費電力をユーザが指定することができる．本評価では BIOS の制約より CPU の電力制御のみを対象とし計測した．

4.2 電力制御ポリシー

オーバープロビジョニング環境における電力管理では，スケジューラがジョブの特性等を考慮し CPU に対する最適な電力割当てを決定することが望ましい．例えば，LLC ミス率や動作周波数の情報を基に最適な電力キャップ値を決定する方法 [28] [29] 等が研究されている．またユーザーが許容可能な性能低下率の情報を与えることで，スケジューラが最適な電力制約値を決定する手法等 [13] もある．

しかしながら，本評価では大規模環境における消費電力や性能の特性を分析することが主な目的であり，特定の電力制御手法に依存しない形で評価を行うことが望ましい．そこで，本評価ではユーザーがジョブの電力キャップ値を与えるものと仮定した．ユーザーはジョブのサブミット時にジョブの実行要求と合わせて電力キャップ値をスケジューラに通知する．将来的にはスケジューラがパフォーマンスカウンタや電力モニタ，プロファイリングツールなどを用いることで，ユーザーからは電力制約の情報は隠蔽可能と考える．スケジューラはユーザーが与えた電力キャップ値を基にスケジューリングを行う．この際，スケジューラはシステム全体の消費電力がシステム全体の電力制約を超えないようにジョブのスケジューリングを行う．ジョブキューの先頭のジョブを実行した際のシステム全体の電力を予測し，予測した電力が制約を超える場合，そのジョブの実行を遅延させる．また，本スケジューリングではバックフィリングを用いてシステムの資源利用率が向上

するように制御を行う．今回は SLURM が提供しているバックフィリングプラグインを用いている．

5. 評価と考察

提案するスケジューラの有効性を確認するため，大規模システムにスケジューラをインストールし様々なテストを行った．初めに，システムの電力制約を変えつつ大規模システムにおいてスケジューラが電力を適切に制御可能かどうかを確認する．さらに，ジョブの実行を待っている遊休ノードの台数に着目しつつ、オーバープロビジョニングシステムの特性につい分析を行う．分析を踏まえたうえで，オーバープロビジョニング環境における追加ノード台数が性能に与える影響について考察する．

5.1 評価内容

本評価ではシステムの電力制約やジョブミックスを変えた際の評価を行う．さらに，それぞれのノード上で行列乗算計算を行うシンセティックなジョブを実行し電力評価を行う．行列乗算は CPU インテンシブな計算であり，高い電力を消費する傾向がある．我々の評価環境のマシンでは CPU ソケットあたり 85W 程度の電力を消費する．評価ではこれらのジョブに対して 85W, 70W, 55W の電力キャップを与えて評価を行う．85W は CPU インテンシブなプログラムを想定している．また，メモリインテンシブなアプリケーションは消費電力が小さい傾向があり，55W のケースはこれらメモリインテンシブなケースを想定したものである．行列乗算はシングルノードジョブであるため，同一の行列計算を複数ノードで実行することで並列プログラムをエミュレーションした．ただし，ノード間の通信は発生しない．我々の目的は提案手法の機能の検証であるため，通信に要する消費電力の検討は本評価には含んでおらず，今後の課題である．

ジョブの到着タイミングやジョブに用いるノード数等の情報は実際の HPC システムのジョブ実行ログを参考にした．本評価では RIKEN RICC スーパーコンピュータ [30] のジョブ実行ログの先頭から 600 個のジョブを用いることとした．

5.2 Validation Under Different System Power Budgets

5.3 システムレベルでの電力制御の検証

初めに HA8000 のすべてのノードを用いてシステムの電力制約を変えつつ，システム全体で消費する電力の計測を行った．システムの電力制約を 100kW, 110kW, 120kW とした場合の実行時の消費電力を計測する．図 2 ではシステムの電力制約を変えた場合のシステム全体の消費電力の推移を示している．

図 2 の結果より提案するスケジューラは電力制約を守りつつジョブの実行を制御することが可能であることが確認できる．システムの負荷が十分に高くなっているグラフ中央部ではシステム全体の消費電力がシステムの電力制約値とほぼ等しくなっている．これらより，提案するスケジューラは大規模環境において電力制約を守りつつジョブを実行可能であり，電力の利用率を改善することができることを示している．

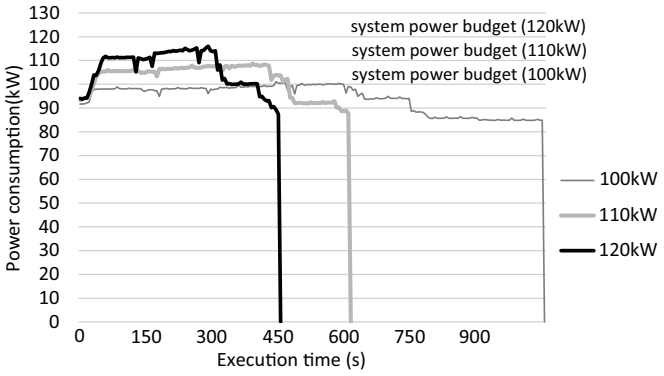

図 2 異なる電力制約下でのシステム全体の消費電力の推移

5.4 消費電力評価

本節ではジョブの到着時刻やジョブミックス，利用するノード数が異なる場合のシステムの消費電力について評価を行う．本評価ではシステムの電力制約値を 500 ノード分の CPU の TDP 電力と等しいものとした．

初めにジョブの到着間隔が異なる場合の電力評価を行う．これは，ジョブの到着間隔が性能に大きな影響を与えるためである．図 3 にシステム全体の実行時の消費電力を示す．到着間隔をオリジナルの 200 倍，400 倍，600 倍と変化させた場合の結果を示している．200 倍は到着間隔が長い場合，400 倍は中程度の場合，600 倍は短い場合を想定している．同じく，ジョブトレースの先頭から 600 個のジョブを用いている．さらに，図では先頭のジョブが投入されてからすべてのジョブが終了するまでの期間の電力を示している．図中の薄い灰色の部分はジョブを実行している計算ノードが消費した電力を示しており，濃い灰色の部分はジョブが割り当てられていないノードが消費した電力を示している．黒い線はジョブキューの中にある実行待ちのジョブ数を示している．

オリジナルのジョブトレースログは本評価で用いるシステムより小さい規模のシステムから得られたログであるため，本評価で用いるような大規模なシステムでは十分に資源を利用することができない．図からわかるように，到着率を 200 倍にしても，多くのアイドルノードがあり，かつ電力にも多くの余剰があることが確認できる．さらに到着間隔を短くするとシステムの消費電力が制限値と同程度に

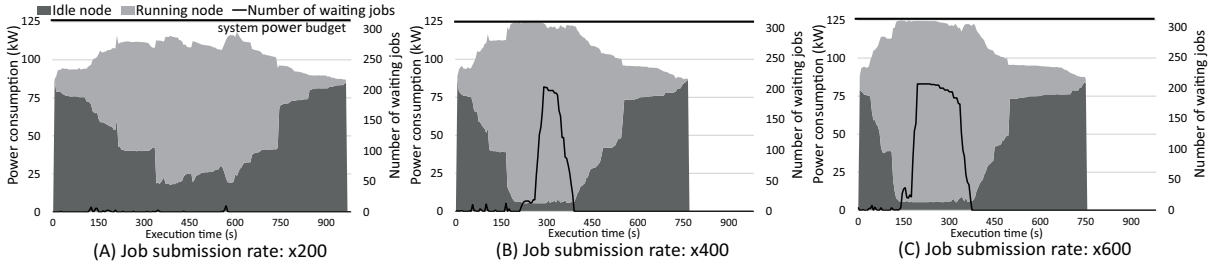

図 3 到着間隔が異なる場合のシステム全体の実行時消費電力の推移

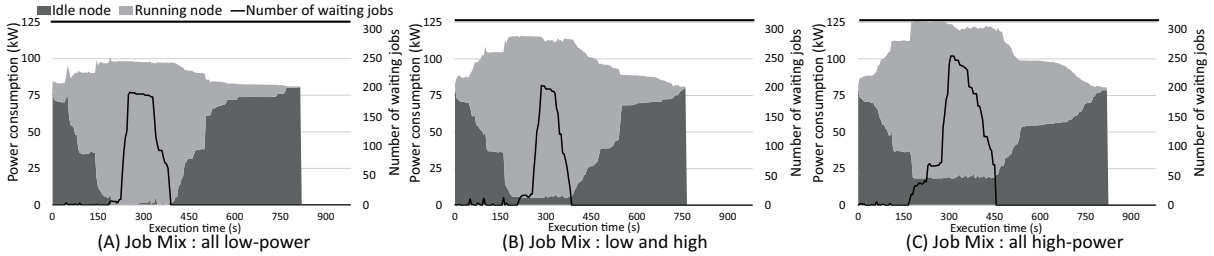

図 4 ジョブミックスが異なる場合のシステム全体の実行時消費電力の推移

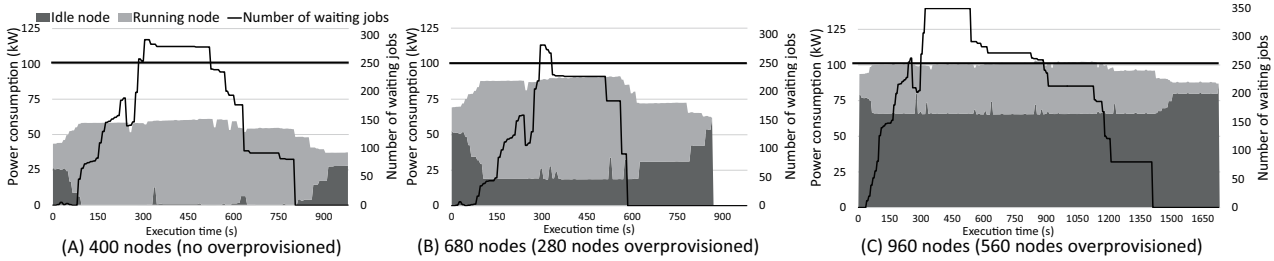

図 5 追加するノード数が異なる場合のシステム全体の実行時消費電力の推移

なり、電力資源を制約ぎりぎりまで使うことができている．到着間隔が 400 倍のケースと 600 倍のケースを比較した場合，電力の利用率は同様であり，さらに，すべてのジョブが終了するまでの時間はほぼ同じであるが，600 倍の際は多くの実行待ちのジョブが生じていることが確認できる．実行待ちジョブが多い場合，実行までの時間が伸びるため以降の実験においては，到着間隔を 400 倍として評価を行うこととした．

図 4 はジョブミックスが異なる場合の評価を示している．消費電力が少ないジョブが多い場合を想定し、電力キャップ値を 55W とした all low-power，消費電力が少ないジョブ (55W) と高いジョブ (85W) が同程度混在している low and high，すべてのジョブの消費電力が高い場合 (85W) を想定した all high-power の 3 種のジョブミックスを比較した．図 4-(A) で示す all low-power において，安定した状態の際（グラフ中央部）はすべてのノードがジョブを実行していることが確認できる．しかしながらシステムの消費電力は制約を大きく下回っている．これは，電力資源が有効に利用できていないことを示している．また，これらの状態は計算ノードが不足しているといえる．一方で，図 4-(C) に示す all high-power の場合，十分な実行待ちのジョブと多数のアイドルノードがあるにも関わらず，システムの消費電力は制約に近い状態となっていることがわかる．これは，電力資源が不足していることを示している．これらの原因はアイドルノードが多くの電力を消費し，新しいジョブの実行を妨げているためである．実際の HPC システムにおいて短い間隔でノードの電源を on/off することは現実的ではないと考えられる．そのため，これらのアイドルノードの消費電力を考慮したシステム設計が重要となる．

図 5 はオーバープロビジョニングする際の追加ノード数を変えた場合の評価結果を示している．図ではシステム全体の消費電力を一定にして追加するノード数を変更した場合の評価結果を示している．図 5 ではシステム全体のノード数を 400 ノード，680 ノード，960 ノードとした場合の結果を示している．また，システム全体の電力制約値は 400 ノードの TDP 分の電力である 100kW とした．すなわち 400 ノードの場合はオーバープロビジョニングしない通常のシステムと同様の過程である。図 5-(A) では大部分のノードがジョブを実行しているにも関わらず，電力に大きな余剰が生じていることが分かる．これは計算ノードが不足していることを示している．5-(B) では 680 ノード，すなわち 280 ノードを追加した場合の結果を示している．400

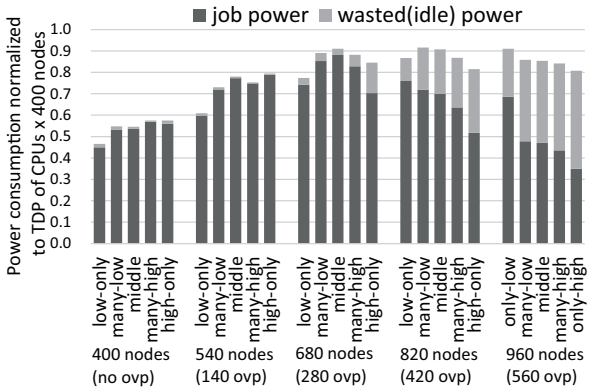

図 6　電力資源の平均利用率

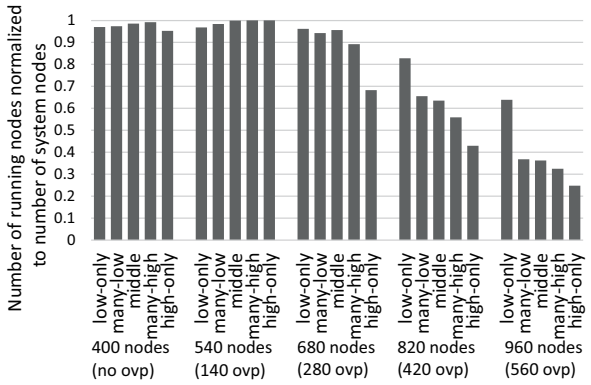

図 7　計算ノード資源の平均利用率

ノードのケースと比較すると全体の実行時間が短くなっており，スループットが改善していることが分かる．これはオーバープロビジョニングによる効果である．また，電力に多少の余剰があるため，さらにノードを追加することで，性能の向上が見込める．5-(C) は 960 ノード，すなわち 560 ノードを追加した場合の結果を示している．このケースでは大部分の電力を消費できていることが確認できる．しかしながら，アイドルノードが消費する電力が大部分を占めていることが分かる．多くのノードが電力不足のため新たなジョブを実行できない．これらの結果より，オーバープロビジョニング環境では追加ノードを増やしすぎるとアイドルノードが消費する電力が増加し，システムの性能が悪化することが分かる．

5.5　オーバープロビジョニング環境における設計指針

前節で示したようにアイドルノードが消費する電力はシステムの性能に大きな影響を及ぼす．アイドルノードの数はジョブミックの特性や追加を行うノード数に依存している．そこで，本節ではアイドルノードの電力に着目する．さらに，これらの分析を元にオーバープロビジョニング環境における追加ノードの決め方について議論をする．はじめに，オーバープロビジョニングされていない 400 ノードの環境を想定し，400 ノードのピーク電力をシステムの電力制約と設定する．さらに，オーバープロビジョニングの度合いを変えた場合を想定し，電力制約を変えないまま，ノード数を 540 ノード，680 ノード，820 ノード，960 ノードとした場合の評価を行う．

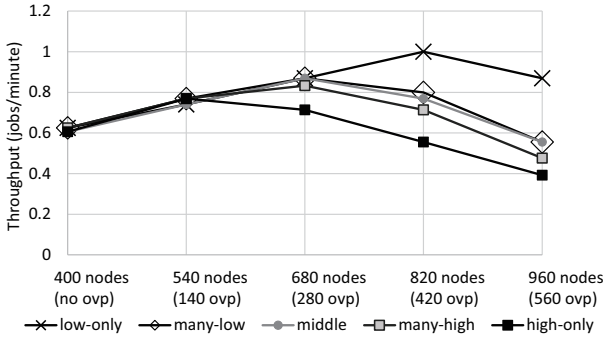

図 8　システムのジョブスループット (jobs/minute)

初めに，様々なジョブミックス下における電力資源利用率，ノード資源利用率，ジョブのスループットをそれぞれ図 6，図 7，図 8 に示す．評価では 5 つのジョブミックスを用いた．これらは，85W:70W:55W = 1:0:0 (high-only)，3:2:1(many-high)，1:1:1 (middle)，1:2:3 (many-low)，0:0:1 (low-only) としている．その他のパラメタは前節と同一である．電力資源利用率は 400 ノードの TDP 電力を 1 として正規化を行っている．また，システムの負荷が安定している時間帯のデータを用いている．

図 6 よりノード数を 680 台まで増やした場合は電力利用率が向上することが分かる．一方でそれ以上ノード数を増やすと逆に電力利用率が低下することが分かる．追加ノードが少ない場合はノード利用率は高い．これは電力を十分に使うことができるためである．一方で追加ノードが多い場合，追加されたアイドルノードが消費する電力が増えることにより，ジョブが利用できる電力が減少している．この結果は図 7 のノード利用率にも示されるように，ノード数が 400 台，540 台の際はおおよそノードの利用率が 1 であるが，それ以上増えるとノードが有効に利用できていないことが分かる．

ジョブスループットについてはノード数が増えるにつれ向上することが図 8 よりわかる．これはオーバープロビジョニングにおいて追加した台数分，有効にジョブを実行できたためである．一方で，960 ノードの場合，スループットは大きく低下している．さらに，低下具合はジョブミックスによって大きく異なることが確認できる．CPU インテンシブなジョブが多いような high-only のようなケースではジョブが多くの電力を消費するため，追加ノードが少ない場合であっても電力資源が枯渇しやすい傾向にある．よって追加ノードは少ないほうが好ましい．一方で

low-only のように個々のジョブの消費電力が少ないメモリインテンシブなジョブが多い場合，電力資源が枯渇しにくいため，多くのノードを追加しても性能が大きく向上する傾向があり，追加ノードは多い方が良いことが分かる．

5.6 オーバープロビジョニング環境の設計ガイドライン

前節までの分析を元にオーバープロビジョニング環境における追加ノード台数の決定方法に関するガイドラインの検討を示す．追加ノードやジョブミックスが異なる際のシステムの性能を，利用可能な CPU 数として求めることとする．ここでは利用可能な CPU 数を CPU_{eff} とする．システムの電力制約 (P_{budget})，システムに導入する CPU 台数 ($CPU_{installed}$)，さらにジョブミックスの特性情報の 3 つから CPU_{eff} を見積もる．ジョブミックスの特性はシステムの導入の際に使用するユーザー層，アプリケーション等から予測可能であるものと仮定する．ジョブミックスは様々なジョブが実行される際のジョブの平均消費電力 P_{avg} を用いることする．

モデルでは初めに，システムの電力制約値より利用可能な CPU($CPU_{available}$) を求める．これは，

$$CPU_{available} = \frac{P_{eff}}{P_{avg}} \quad (1)$$

として求める．また，P_{eff} はジョブ実行によって有効に消費される電力値を示している．P_{eff} は CPU の静的な電力値 P_{base} を用いて次のように計算することができる．

$$P_{eff} = P_{budget} - (P_{base} \times CPU_{installed}) \quad (2)$$

最後に有効な CPU 数を求める．導入した数以上の CPU は利用できないため， CPU_{eff} は次に示すように，

$$CPU_{eff} = min(CPU_{available}, CPU_{installed}) \quad (3)$$

となる．

図 9 に導入するノード数とジョブミックスを変えた場合の有効な CPU 数 CPU_{eff} を示している．P_{avg} は単純に実行中のジョブの平均消費電力をベースにした．この結果は HA8000 で実際に計測を行ったジョブスループットの図 8 と同じ形をしていることが確認できる．そのことより，本モデルによってシステムの性能を設計時に予測できるものと考えている．オーバープロビジョニング環境において，実運用されるであろうジョブの平均消費電力が予測できればどれだけの追加ノードを購入すべきかを予測することが可能となる，一方で本モデルは DRAM の電力やファンの電力，ネットワークの電力等を含んでいない．そのため，これらの考慮を行うことが今後の課題である．

6. まとめ

余剰電力を有効に使うための方法としてハードウェアオーバープロビジョニングが着目されているが，多くの研究はシミュレーションによる評価を用いたり，小規模な環境を想定しているため大規模な実環境に対応し難い課題があった．本研究ではこれらの課題を解決するために新たなリソースマネージャの構成について示した．さらに，提案するスケジューラを大規模環境に対しインストールし評価を行った．これらの結果より追加ノード数とジョブミックスが性能に与える影響を明らかとした．

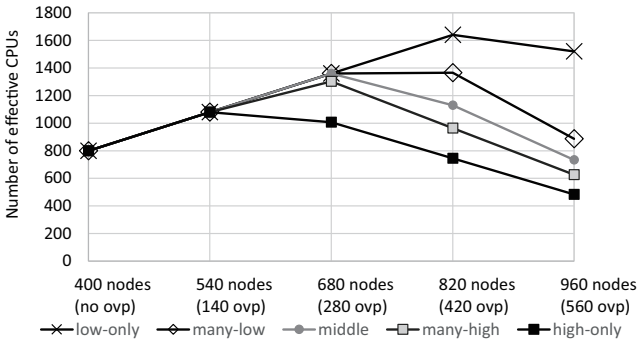

図 9 利用可能な CPU 数の予測

謝辞 本研究は科学技術振興機構・戦略的創造研究推進事業 (CREST) の研究プロジェクト「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」の助成により行われたものである．また，本研究の一部は，ローレンス・リバモア国立研究所の米国エネルギー省の助成により行われたものである．

参考文献

[1] Strohmaier, E., Dongarra, J., Simon, H. and Meuer, M.: Top500 Supercomputer Sites (2014).
[2] Mu'alem, A. W. and Feitelson, D. G.: Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling, *IEEE Trans. Parallel Distrib. Syst.*, Vol. 12, No. 6, pp. 529–543 (online), DOI: 10.1109/71.932708 (2001).
[3] Etinski, M., Corbalan, J., Labarta, J. and Valero, M.: Utilization Driven Power-aware Parallel Job Scheduling, *Computer Science - R&D*, Vol. 25, No. 3-4, pp. 207–216 (2010).
[4] Etinski, M., Corbalan, J., Labarta, J. and Valero, M.: Optimizing Job Performance Under a Given Power Constraint in HPC Centers, *Green Computing Conference*, pp. 257–267 (2010).
[5] Etinski, M., Corbalan, J., Labarta, J. and Valero, M.: Linear Programming Based Parallel Job Scheduling for Power Constrained Systems, *International Conference on High Performance Computing and Simulation*, pp. 72–80 (2011).
[6] Etinski, M., Corbalan, J., Labarta, J. and Valero, M.: Parallel Job Scheduling for Power Constrained HPC Systems, *Parallel Computing*, Vol. 38, No. 12, pp. 615–630 (2012).
[7] Borghesi, A., Conficoni, C., Lombardi, M. and Bartolini, A.: MS3: A Mediterranean-stile job scheduler for supercomputers-do less when it's too hot!, *High Performance Computing & Simulation (HPCS), 2015 International Conference on*, IEEE, pp. 88–95 (2015).

[8] Gholkar, N., Mueller, F. and Rountree, B.: Power Tuning HPC Jobs on Power-Constrained Systems, *Proceedings of the 2016 International Conference on Parallel Architectures and Compilation*, PACT '16, New York, NY, USA, ACM, pp. 179–191 (online), DOI: 10.1145/2967938.2967961 (2016).

[9] Patki, T., Lowenthal, D. K., Sasidharan, A., Maiterth, M., Rountree, B. L., Schulz, M. and de Supinski, B. R.: Practical Resource Management in Power-Constrained, High Performance Computing, *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '15, New York, NY, USA, ACM, pp. 121–132 (online), DOI: 10.1145/2749246.2749262 (2015).

[10] Sarood, O., Langer, A., Gupta, A. and Kale, L.: Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, Piscataway, NJ, USA, IEEE Press, pp. 807–818 (online), DOI: 10.1109/SC.2014.71 (2014).

[11] Ellsworth, D., Malony, A., Rountree, B. and Schulz, M.: POW: System-wide Dynamic Reallocation of Limited Power in HPC, *High Performance Parallel and Distributed Computing (HPDC)* (2015).

[12] Ellsworth, D., Malony, A., Rountree, B. and Schulz, M.: Dynamic Power Sharing for Higher Job Throughput, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, New York, NY, USA, ACM, pp. 80:1–80:11 (online), DOI: 10.1145/2807591.2807643 (2015).

[13] Cao, T., He, Y. and Kondo, M.: Demand-Aware Power Management for Power-Constrained HPC Systems, *The 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGrid'16 (2016).

[14] Ellsworth, D., Patki, T., Perarnau, S., Seo, S., Amer, A., Zounmevo, J., Gupta, R., Yoshii, K., Hoffman, H., Malony, A., Schulz, M. and Beckman, P.: Systemwide Power Management with Argo, *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 1118–1121 (online), DOI: 10.1109/IPDPSW.2016.81 (2016).

[15] Borkar, S., Karnik, T., Narendra, S., Tschanz, J., Keshavarzi, A. and De, V.: Parameter Variations and Impact on Circuits and Microarchitecture, *Proceedings of the 40th Annual Design Automation Conference*, DAC '03, New York, NY, USA, ACM, pp. 338–342 (online), DOI: 10.1145/775832.775920 (2003).

[16] Hong, S., Narayanan, S. H. K., Kandemir, M. and Özturk, O.: Process Variation Aware Thread Mapping for Chip Multiprocessors, *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, 3001 Leuven, Belgium, Belgium, European Design and Automation Association, pp. 821–826 (online), available from ⟨http://dl.acm.org/citation.cfm?id=1874620.1874821⟩ (2009).

[17] Totoni, E., Langer, A., Torrellas, J. and Kale, L.: Scheduling for HPC Systems with Process Variation Heterogeneity (2015).

[18] Inadomi, Y., Patki, T., Inoue, K., Aoyagi, M., Rountree, B., Schulz, M., Lowenthal, D., Wada, Y., Fukazawa, K., Ueda, M., Kondo, M. and Miyoshi, I.: Analyzing and Mitigating the Impact of Manufacturing Variability in Power-constrained Supercomputing, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, New York, NY, USA, ACM, pp. 78:1–78:12 (online), DOI: 10.1145/2807591.2807638 (2015).

[19] Downey, A. B.: A Model For Speedup of Parallel Programs, Technical report, Berkeley, CA, USA (1997).

[20] Georgiou, Y. and Hautreux, M.: Evaluating scalability and efficiency of the Resource and Job Management System on large HPC Clusters, *Workshop on Job Scheduling Strategies for Parallel Processing*, JSSPP '12 (2012).

[21] Bodas, D., Song, J., Rajappa, M. and Hoffman, A.: Simple Power-aware Scheduler to Limit Power Consumption by HPC System Within a Budget, *Proceedings of the 2Nd International Workshop on Energy Efficient Supercomputing*, E2SC '14, Piscataway, NJ, USA, IEEE Press, pp. 21–30 (online), DOI: 10.1109/E2SC.2014.8 (2014).

[22] Jette, M.: Slurm Power Management Support, `https://slurm.schedmd.com/SLUG15/Power_mgmt.pdf` (2015).

[23] Borghesi, A., Bartolini, A., Lombardi, M., Milano, M. and Benini, L.: Predictive Modeling for Job Power Consumption in HPC Systems, *High Performance Computing: 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, June 19-23, 2016*.

[24] David, H., Gorbatov, E., Hanebutte, U. R., Khanna, R. and Le, C.: RAPL: Memory Power Estimation and Capping, *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '10, New York, NY, USA, ACM, pp. 189–194 (online), DOI: 10.1145/1840845.1840883 (2010).

[25] Intel: *Intel-64 and IA-32 Architectures Software Developer's Manual, Volumes 3A and 3B: System Programming Guide* (2011).

[26] : `http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-home.html`.

[27] Broyles, M., Francois, C., Geissler, A., Grout, G., Hollinger, M., Rosedahl, T., J. Silva, G., Vanderwiel, M., Van Heuklon, J. and Veale, B.: IBM EnergyScale for POWER7 Processor-Based Systems (2011).

[28] Schöne, R. and Hackenberg, D.: On-line Analysis of Hardware Performance Events for Workload Characterization and Processor Frequency Scaling Decisions, *Proceedings of the 2Nd ACM/SPEC International Conference on Performance Engineering*, ICPE '11, New York, NY, USA, ACM, pp. 481–486 (online), DOI: 10.1145/1958746.1958819 (2011).

[29] Weissel, A. and Bellosa, F.: Process Cruise Control: Event-driven Clock Scaling for Dynamic Power Management, *Proceedings of the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, CASES '02, New York, NY, USA, ACM, pp. 238–246 (online), DOI: 10.1145/581630.581668 (2002).

[30] : The RICC log, `http://www.cs.huji.ac.il/labs/parallel/workload/l_ricc/index.html` (2010).