Vol.2016-HPC-155 No.20
2016/8/9

電力制約を考慮した資源管理ツールによるHPCシステムの電力性能解析

坂本 龍一[†1,†4,a)]　カオ タン[†1,†4]　和 遠[†1,†4]　近藤 正章[†1,†4]　深沢 圭一郎[†2]　上田 将嗣[†3]
稲富 雄一[†3,†4]　井上 弘士[†3,†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電力制約適応型システムが注目されている．このような HPC システムでは，電力制約を考慮したノードの資源管理やジョブスケジューリングを行い，さらにはシステムの資源管理ソフトウェアから電力ノブを適切に制御することが必須となる．従来より電力制約を考慮した資源管理の研究は多数あるが，大規模システムを意識した資源管理ツールの開発や評価は十分に行われてはいない．そこで本稿では，我々が開発中の電力制約を考慮した資源管理ソフトウェアを用いてジョブスケジューリングや計算ノードの電力ノブ制御を行い，大規模システム上で多数のジョブを実行した際の性能・電力消費特性を解析した結果について示す．

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．たとえば，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 15 MW 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電力制約適応型システムがポストペタスケール HPC システムのあるべき姿との認識に立ち，その実現に必要となる電力マネージメントフレームワークの研究を進めている [2]．

このような電力制約適応型システムでは，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電力マネージメントフレームワークの重要な役割の 1 つとなる．従来の計算ノードの資源管理方法は，主にジョブが利用するノード数に着目し，遊休計算機資源が最小化されるようにジョブを選択することで，計算効率を向上させるアプローチをとっていた．しかし，この場合，各計算ノードが最大電力を利用することはまれであり，結果的に電力に余剰が生じ電力資源も含めた資源の有効利用が行えているとは言えない．一方でジョブの使用電力を考慮したうえで計算ノードの利用を決定することにより，システム全体の実行効率を向上させることが可能である．

たとえば，メモリインテンシブなジョブは CPU の利用効率が低く，CPU インテンシブなジョブと比較すると，計算ノードあたりの消費電力が低くなる．一方で，CPU インテンシブなジョブでは消費電力が大きくなる傾向がある．そのため，これらのジョブ毎の消費電力とシステム全体の余剰電力を考慮し，実行するジョブを選択することにより実行効率を改善することができる．具体的には，ジョブのスケジューリングを行う際に，消費電力が電力制約を超えない範囲でジョブを選択・投入することが必要となる．一

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

方でこのようなシステムでは，ジョブの電力特性やジョブが利用するノード数，ジョブの実行時間，ジョブの到着時刻等がシステムの特性を左右する．

従来から，電力制約下での電力資源管理に関する研究は行われている．例えば，Patki らはジョブの実行時間をプロファイリングにより予測し，最適なジョブサイズと電力制約値をスケジューラが調整することで，電力利用率の改善を行っている [3]．また，Cao らは実行時にパフォーマンスカウンタの値を参考にしつつ，実行時間の予測を行うことで，最適な電力キャップ値を算出する手法を提案している [4]．一方で，これらの研究では大規模な HPC システムでの評価は十分に行われておらず，実際の HPC システム，また実際のリソースマネージャを用いて電力制約適応型システムの電力・性能特性を解析することが重要である．

そこで，本稿では我々が開発中の電力制約適応型リソースマネージャを用い，電力制約を考慮し資源管理を行った際の電力・性能特性について調査を行う．具体的には，実際の HPC システムを用いて様々なジョブセットを実行した場合の電力効率について評価する．

2. 電力制約を考慮したリソースマネージャ

2 章では従来のリソースマネージャの課題を示し，電力制適応約型システム向けリソースマネージャの要件についてまとめる．

2.1 従来のリソースマネージャ

リソースマネージャは HPC システムを利用するユーザーからのジョブ実行要求を受け，計算ノード上で各ジョブを実行するものであり，2 つの大きな役割がある．1 つ目は，ユーザーからの多数のジョブ実行要求を管理するジョブスケジューラである．複数のジョブ間での開始時刻や実行期間・優先度などを考慮し，ジョブの実行順序を最適化する．将来的なジョブ実行の計画を立て，計算ノードに余りがある場合は，後から投入されたジョブを前倒しして行うバックフィリング [5] などを行う．2 つ目はジョブと計算ノードの対応を管理するノードスケジューラである．従来のスケジューラは主にジョブからの要求計算ノード数を考慮し，利用する計算ノードを決定している．

2.2 電力制約適応型システムにおける資源管理

従来の HPC システムでは，全計算ノードが最大の負荷となり，最も電力を消費した場合の最大消費電力が消費電力制約を超えないようにハードウェア資源が設置されていた．しかし，計算ノードが最大の負荷となるジョブはまれであり，電力資源に余剰がある場合が多い [6][7]．そこで，電力制約適応型システムでは，積極的に電力制約を超えるようなハードウェアを投入し，ハードウェア資源の使用量やハードウェアが持つ電力ノブを最適化することで，電力余剰を最小限にし実行効率を高めるアプローチをとる．

そのため，電力制約適応型システムにおけるリソースマネージャは，電力を考慮したノードスケジューリングと計算ノードの電力ノブの調整も管理する必要がある．具体的にはジョブが消費する電力，全計算ノードの利用状況，そして電力制約を考慮したうえで，ジョブと計算ノードの割り当てを管理することで，電力制約の中で電力資源を最大限利用できるようにしつつ，実行性能の向上を図る．また，計算ノード全体の管理とともに個々の計算ノードの電力管理も行う．個々の計算ノードの電力ノブを制御し，システム全体の電力制約を超えないように制御することが重要である．

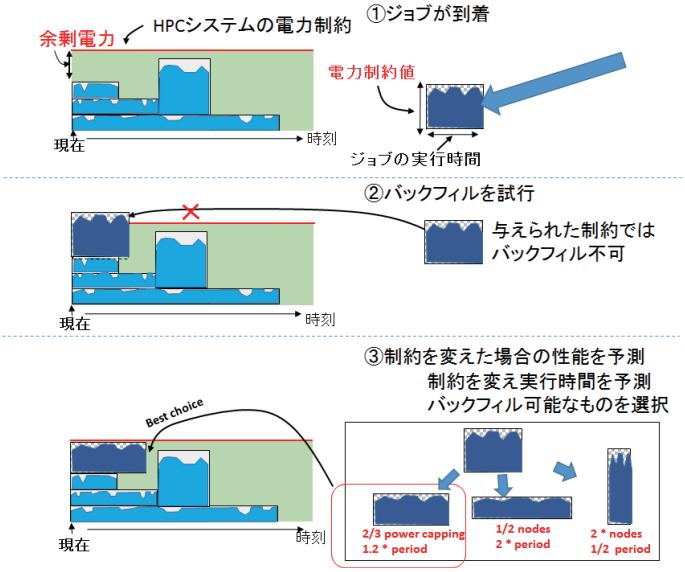

図 1 電力制約適応型システムにおけるバックフィリング

また，電力制約適応型システムでは，電力効率の向上を図るために，ジョブに対する電力制約値を変えた場合の実行時間を予測し，バックフィリングを行う．その概要を図を 1 に示す．

2.3 ジョブの電力特性

実際のジョブを実行した場合，多くの場合で計算ノードの消費電力は設計最大消費電力を下回る．ジョブの実行においてボトルネックとなる部分があり性能が制約されることから，ハードウェア資源をフルに利用できないためである．したがって，ジョブの性能特性に依存して電力消費特性も大きく異なる．一般にメモリインテンシブなアプリケーションや通信インテンシブなアプリケーションは電力制約を与えた場合の性能低下率が小さく，CPU インテンシブなアプリケーションは電力制約を与えると性能が大きく低下する傾向にある．

電力制約適応型システムではこれらのジョブの電力特性を考慮することが重要となる．これらの特性をスケジュー

リングに用いることで，電力資源を効率的に利用できると考えられる．

2.4 ジョブの特性を表す電力ヒント

電力制約適応型システムでは，効率的なスケジューリングのためには，各ジョブが要求する消費電力の情報を参考することが重要である．この情報を得る手段として、下記の 3 つが考えられる．

- ユーザーによる電力ヒント情報の付加：
  ジョブを発行するユーザーが，ジョブの電力特性を理解していることを期待し，ジョブの実行と合わせて電力ヒント情報を明示的に付加するものである．ユーザーによる電力ヒントとしては周波数や電力キャップ値などが考えられる．ジョブが利用できる電力を制約することによりジョブの性能が低下する恐れがあるため，ジョブの性能をできるだけ落とさない範囲で電力制約を決定することが望ましい．
  一方で，ジョブの性能低下を許容し，厳しい電力制約を与える場合も考えられる．計算ノード全体としての電力余剰を極力なくし，電力制約内での実行効率を最大化する場合に有効である．ジョブに対して低い電力制約を与え，全体の電力制約の中でなるべく多くのジョブを実行させることができれば，システムの実行効率を向上させることができる．
- 静的なプロファイリング：
  ジョブのプログラムを静的に解析し，最適な電力ヒント情報を自動的に抽出する手法である．静的なプロファイリングには実行前に解析を行うものや [8]，過去のジョブ実行履歴をベースに電力特性を抽出する手法がある [9]．コンパイラ等で自動的に解析を行うことでユーザーがジョブの電力特性を意識する必要はない．しかし，実環境で生じる動的な要因へ対応できないことや，プロファイリングに要する時間的なオーバヘッドが課題である．
- 動的なモニタリング：
  ジョブ実行をリアルタイムにモニタリングし，電力ヒント情報を得る方法である．計算ノードの電力モニタや各種カウンタの情報を用い，動的に電力ノブの調整を行う手法も提案されている [10]．動的に決定されるプログラムの挙動に対応可能であるが，数万台規模のような多くの計算ノードを持つシステムでは，リアルタイムの制御が難しいという課題もある．

2.5 電力モニタリングと電力ノブ制御

電力制約適応型システムでは消費電力のモニタリングとジョブに対する電力制約の設定が重要である．電力モニタリングに関しては，計算機内部で電流計を用いて計測する手法と，プロセッサの性能カウンタの値を参考に電力値を予測する方法がある．前者は例えば IPMI を用いた計測手法が提供されており，CPU 電力，メモリ電力，ノード全体の電力，筐体温度，ファン速度等を取得することができる環境が存在する．後者は，Intel が提供している RAPL[11][12] が良く用いられる．また，電力制約の設定にも RAPL が用いられることが多い．RAPL は Intel の Sandy Bridge 以降の Xeon プロセッサに搭載された電源管理機構であり，ソケット単位での電力制約を指定することができる．設定後，プロセッサは自動的に DVFS 等の省電力技術を行うことにより，電力制約を大きく超えることなく実行が行われる．

3. 電力制約適応型システムの電力特性解析手法

電力制約適応型システムでの性能・電力消費特性は，個々のジョブ特性やジョブ到着率に大きく影響される．そのため，これらを変化させつつシステムの特性を評価する必要がある．

HPC システムは多数のユーザーが利用し，各ユーザが実行するジョブの特性は大きく異なる．全てのジョブを入手，運用中のシステムでそれらを実行して解析することは現実的ではない．そこで，ジョブのもともとの消費電力や，電力制約をかけた際の実行時間について，特性の異なるジョブをエミュレートするような仕組みが必要となる．本稿ではキャッシュミス率を制御し，CPU ／メモリインテンシブネスを変化させることで，様々なジョブの特性をエミュレートする．

例えば，キャッシュミス率が小さい場合，すなわち CPU インテンシブなジョブの場合，電力制約を与えると CPU の動作周波数が低下するため，それに応じ大きく性能が低下する傾向がある．一方で，キャッシュミス率が高い場合，すなわちメモリインテンシブなジョブの場合，メモリアクセス時間が支配的であるため，CPU の周波数を下げても性能低下が小さい傾向がある．

4. 電力を考慮した資源管理の実装

我々は Slurm リソースマネージャ [13] を拡張する形で，電力制約を考慮したリソースマネージャの実装を行っている．Slurm は Top500 の中でも最も多く利用されているリソースマネージャであり，電力制約を考慮した資源管理機構を実装する有用性は高いと考えられる．

4.1 全体構成と電力考慮のための拡張

本実装では Slurm の拡張と合わせ，計算ノードの電力を管理するパワーマネージャを導入した．これらの実装を図 2 に示す．今回は，ジョブの実行要求と合わせてジョブの電力ヒントを与える実装とした．スケジューラは与えられたジョブ情報と電力ヒント，計算ノード全体の電力制約を

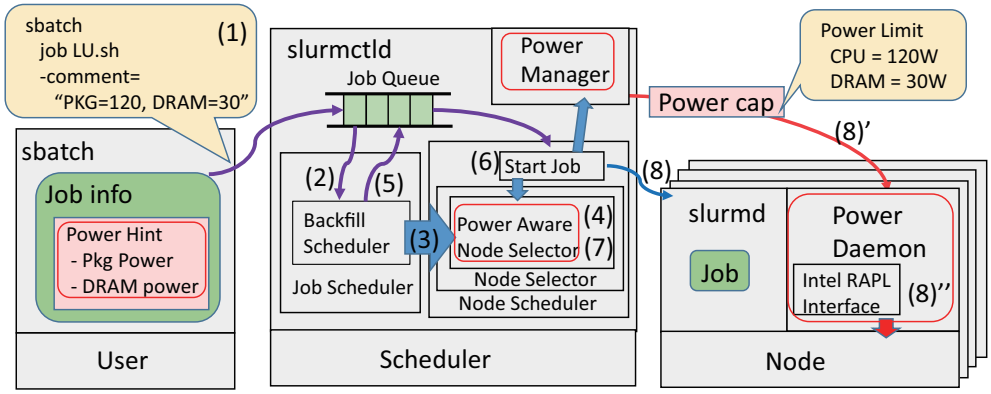

図 2　電力を考慮した slurm リソースマネージャ

考慮したうえでジョブスケジューリングを行う．さらに，個々の計算ノードでは割り当てられたジョブを行うとともに，与えられた電力ヒントを用いて電力管理を行う．図 2 では電力制約を考慮したリソースマネージャを実現するために追加を行った部分を赤枠で示しており，電力ヒントインタフェース，電力モニタリング機能，計算ノード上の電源管理デーモンが追加されている他，スケジューラも改変を行っている．

4.2　電力を考慮したスケジューラ拡張

計算ノードの管理はスケジューラ内にあるノードスケジューラで行われるため，当該部分を拡張した (図 2 中央赤枠内)．まず，ノードスケジューラは，利用可能な計算ノード数とジョブが要求する計算ノード数を比較し，ジョブが実行可能であるかを確認するが，このノードセレクタに対し電力制約を考慮するように拡張を行った．具体的には，実行中のジョブ情報から全計算ノードの消費電力を算出し，新たに割り当てを行うジョブを実行するための余剰電力があるかを確認する機能を追加した．本拡張部は，ジョブキューでのバックフィリングとジョブの計算ノード割り当てに用いられる．

また，全計算ノードの電力を管理するパワーマネージャ (図 2 中央上側) を導入した．パワーマネージャはジョブ実行の直前に，ジョブが利用する計算ノード情報と電力制約情報をスケジューラから受け取り，該当する計算ノードに対し電力制約を行うよう依頼する．個々の計算ノードには電力制約に従い電力ノブの調整を行うパワーデーモン (図 2 右側) が導入される．パワーデーモンはパワーマネージャから与えられた電力制約に従い，計算ノードが電力制約を超えないように電力ノブを調整する．このように，個々の計算ノードが電力制約を守ることによって，システム全体で電力制約を守ることができる．

本リソースマネージャ内部の連携について詳細を以下で述べる．

(1) ジョブの投入と電力制約の付加（図 2(1)）
ジョブを投入する際にジョブのヒントを計算ノードが利用する最大消費電力として指定する．sbatch のコメント機能を用いて CPU 電力制約，DRAM 電力制約を付加する．

(2) バックフィリングのためのジョブ情報の探索 (図 2(2))
バックフィリングの際はジョブキューを探索し，ジョブキューの最適化を行う．バックフィルスケジューラは特定のジョブを 1 つ選択し，最適化が可能であるかを確認する．ジョブキュー内のすべてのジョブ情報に対し探索を行い，ジョブキュー全体の最適化を行う．これらは定期的に繰り返される．

(3) ジョブ開始予想時刻の問合せ（図 2(3)）
バックフィリングは，ジョブの開始時刻を見積もることでジョブ情報の入替えを行う．具体的には，選択したジョブについて，計算ノードの利用状況を考慮したうえで開始可能時刻をノードスケジューラに対し問い合わせる．

(4) ジョブ開始予想時間の算出 (図 2(4))
ノードスケジューラは計算ノードの構成や利用状態を考慮し，ジョブの開始可能時刻を算出する．拡張を行ったノードセレクタでは，計算ノード数の確認と合わせて電力の余剰についても確認を行う．

(5) ジョブキュー内のジョブ情報の入替え (図 2(5))
ジョブスケジューラはジョブ開始可能時刻の見積もりを受け，ジョブキュー内部の順番を修正する．これによって，計算ノードの利用効率を向上させる．

(6) 計算ノードで実行するジョブの決定 (図 2(6))
計算ノードで実行されるジョブの決定は，ノードスケジューラがジョブキューの先頭のジョブを取り出すことによって行われる．ジョブ実行の際も，(7) に示すようにジョブの要求する計算ノード数・電力が満たされているかを確認する．

(7) ジョブ実行直前の電力余剰の確認 (図 2(7))
ジョブ実行の直前にも，再度電力やノード数に余剰があることを確認する．この際も，追加を行ったノード

セレクタによって計算ノード数と余剰電力の確認を行う．

(8) ジョブの実行と計算ノードの電力制御
計算ノード数と電力制約の条件を満たしたジョブは計算ノードの slurmd デーモン上で実行される．合わせて，パワーマネージャは割り当てられた計算ノードに対し，電力制約を通知する (図 2(8)')．さらに，計算ノードではパワーデーモンが電力制約を受け，各計算ノードで利用できる電力を守るように電源制御を行う (図 2(8)")．各計算ノードでは電力制約に合わせ，ハードウェアの電源ノブを調整することで，電力制限を超えないようにする．なお，本実装では RAPL を用いている．

5. 評価

本章では開発したリソースマネージャを用い，様々な特性を持つジョブを実行した場合を想定しつつ，電力制約適応型システムにおける性能・電力特性について評価結果を示す．まず，ジョブの電力特性の実装と評価について述べ，その後に実際の HPC システムと 4 章で述べた slurm スケジューラを用いて性能・電力特性を解析する．最後に，RAPL と IPMI による電力モニタリングの精度について考察する．

表 1　電力特性評価用ノード

Processor	Intel Xeon E5-2670 v2	(2 scokets)
	Num. of Cores	10
	Frequency	2.5Ghz
	L1 Cache	32KB I + 32KB D per core
	L2 Cache	256KB per core
	L3 Cache	25MB per chip
DRAM	Size	64GB

5.1 ジョブの電力特性のエミュレーション

ジョブの電力特性のエミュレーションのため，行列計算のサイズを変更してキャッシュミス率を調整可能なジョブを作成した．本ジョブは OpenMP を用いてノード内の並列化をしている．

本ジョブに対し，行列サイズと電力キャップ値を変えた場合の実行時間について測定を行った．なお，本評価は表 1 に示すノードを用いた．行列サイズを 512 x 512 から 8192 x 8192 まで変更し，L3 キャッシュにデータが乗る場合と乗らない場合を考慮した．また，電力制約の最大値と最小値は，それぞれ RAPL がサポートする 55W と 110W を用い，この間を 5W 刻みで設定して実行時間の変化を計測した．これらの結果を図 3 に示す．

評価結果より，行列サイズが 512,1024 の場合，電力制約を課すと 60%以上実行時間が伸びることがわかった．これらは CPU インテンシブな場合であり，電力制約を与えることで CPU の動作周波数が低下し，性能が低下したためである．一方で 2048 の場合，RAPL がサポートする最小値である 55W の電力キャップ値を与えても，10%程度しか性能が低下しないことがわかった．さらに，サイズの大きな 4096 や 8192 の場合，電力キャップを最小の 55W に設定しても性能がほぼ低下しなかった．これらは，メモリアクセス時間が支配的であり，演算性能が実行時間にあまり影響しないためと考えられる．

表 2　HA8000 の仕様

Node 台数	965 台	
Processor	Intel Xeon E5-2697 v2	(2 scokets)
	Num. of Cores	12
	Frequency	2.7Ghz
	L1 Cache	32KB I + 32KB D per core
	L2 Cache	256KB per core
	L3 Cache	30MB per chip
DRAM	Size	256GB

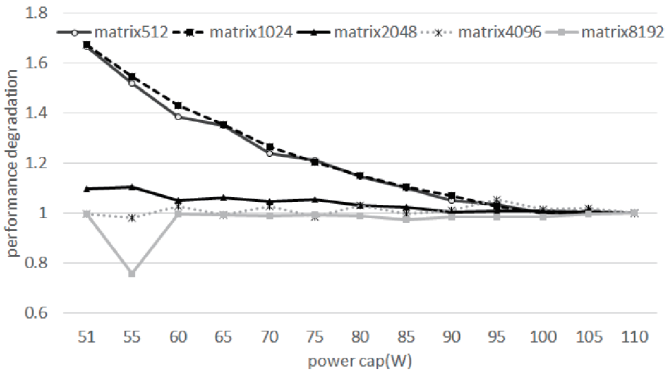

図 3　電力制約とアプリケーションの実行時間

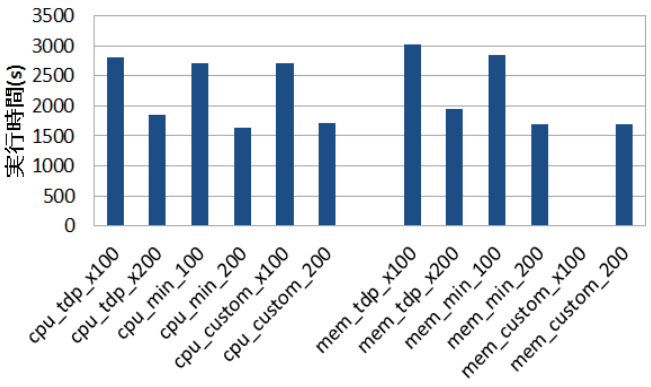

図 4　実行時間

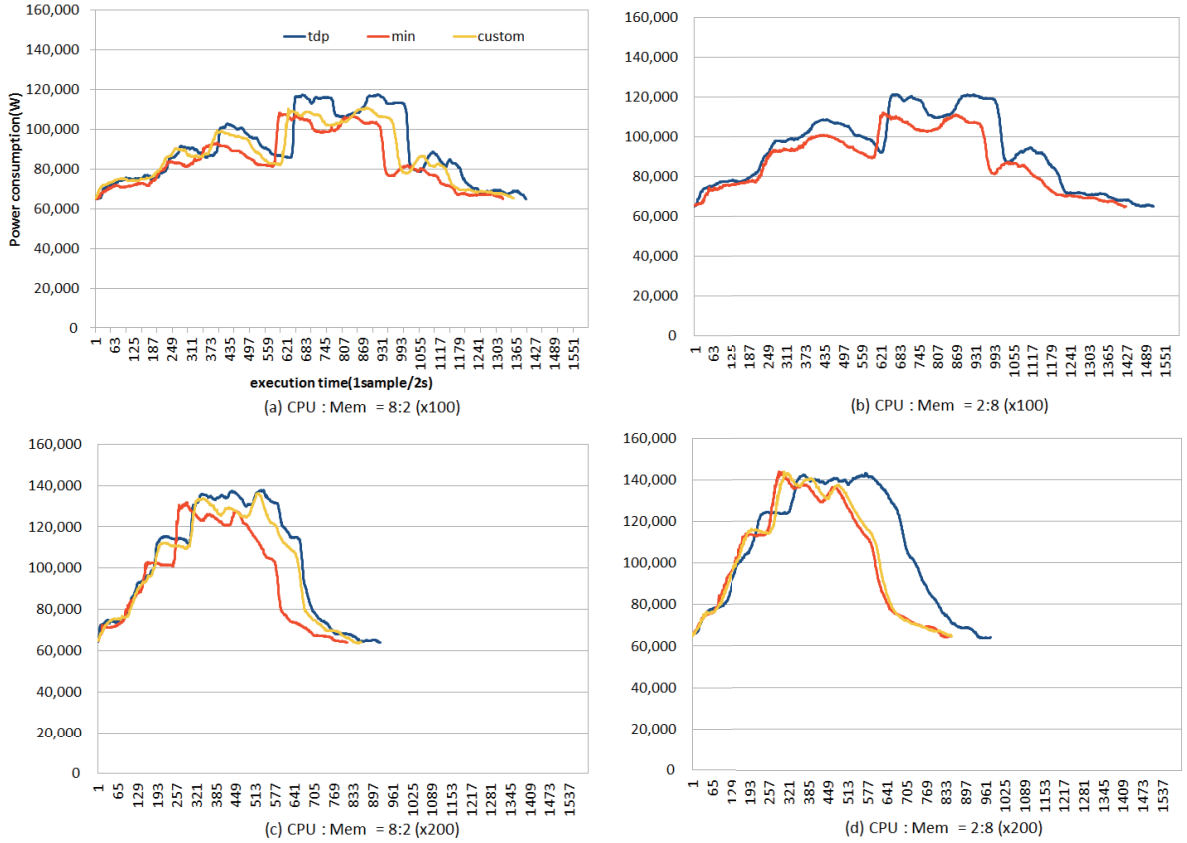

図 7　実行中の HPC システムの消費電力

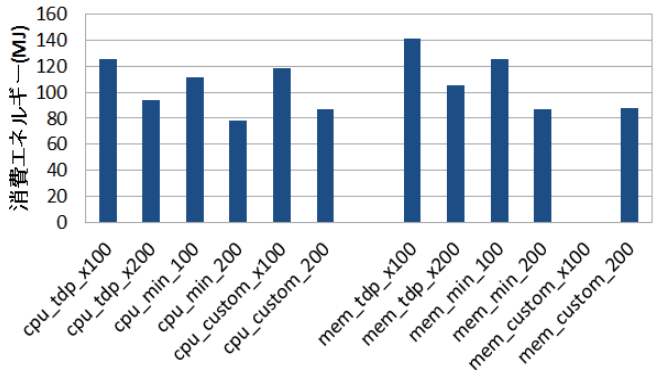

図 5　消費エネルギー

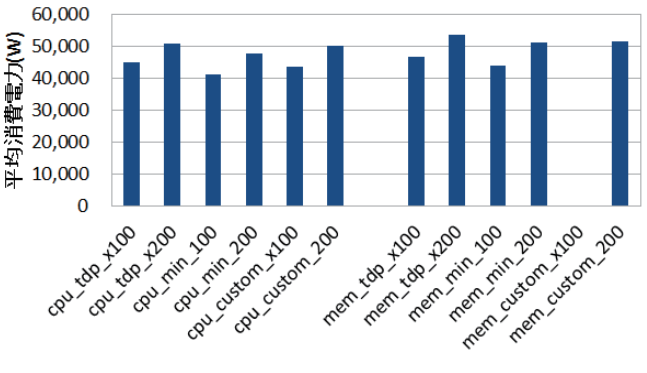

図 6　平均消費電力

5.2　システムの特性

5.1 節のジョブを利用し，九州大学の HA8000 システムを用いて消費エネルギー，平均消費電力，ジョブの実行時間の評価を行った．HA8000 の詳細を表 2 に示す．オーバープロビジョニング環境を想定し，965 台のノードの内，7 割のノードが最大の電力を消費した場合の電力をシステムの電力制約値とした．なお，216KW の電力に相当する．

ジョブの到着タイミングには，過去の HPC システムのジョブ投入ログを用いる．具体的には，理研 RICC の 2010 年のログ [14] を用い，その先頭から 600 ジョブを実行すると想定する．

ジョブ特性の違いによる性能・電力消費特性の違いを解析するため，600 個のジョブに対しその特性に偏りを持たせた．具体的には 600 ジョブのうち 8 割が CPU インテンシブなジョブ (行列サイズ 512) で 2 割がメモリインテンシブ (行列サイズ 8192) な場合と，2 割が CPU インテンシブなジョブで 8 割がメモリインテンシブな場合の 2 つのケースを評価する．システム規模の違いを考慮し，ジョブの到着間隔を短くしてノードの利用率が上がるようにした．本評価では，ジョブの到着間隔をログにある到着間隔の 100 分の 1，あるいは 200 分の 1 に変更した．

比較では，電力キャップ値を TDP に設定して電力制約を行わない場合 (tdp)，実験で用いた CPU がサポートす

る最小の電力キャップ値である 55W に固定的に設定した場合 (min)，CPU インテンシブなジョブに対しては電力制約を与えずメモリインテンシブなジョブにのみ 55W の電力キャッピングを与えた場合 (custom) の 3 通りについて評価を行う．

600 ジョブがすべて終了するまでの時間を図 4 に，消費エネルギーの結果を図 5 に，平均消費電力を図 6 に示す．また，ジョブ実行中の消費電力の変化を図 7 に示す．グラフ内の各項目の名前は，CPU インテンシブなジョブが多いかメモリインテンシブなジョブが多いか (cpu/mem)，そして電力制約 (tdp/min/custom)，最後に到着間隔 (x100/x200) を示している．例えば，cpu インテンシブなジョブが 8 割，電力制約には TDP を用い，到着間隔を 100 分の 1 に設定した場合は cpu_tdp_x100 と表されている．なお，mem_custom_x100 はデータ取得が間に合わなかったため示していない．

評価の結果，600 ジョブをすべて実行し終わるまでの時間は cpu_min_200 が最も短い結果となった．同一条件においてスケジューリングに tdp を用いた結果と比較し，11%性能を改善することができた．これは，min ではプログラムの実行時間が伸びるものの，それ以上に同時に実行できるジョブ数が増えたことが原因と考えられる．また，到着率を x100 から x200 にすると消費エネルギーが削減された．これは，余剰ノードが有効に利用されたためと考えられる．

図 7 を見ると最大消費電力は 140,000W 程度であることが確認できる．システムの電力制約は 217,000W であり，まだ電力に余剰があるが，この原因として割当てるべきジョブやノードが不足していることが考えられる．

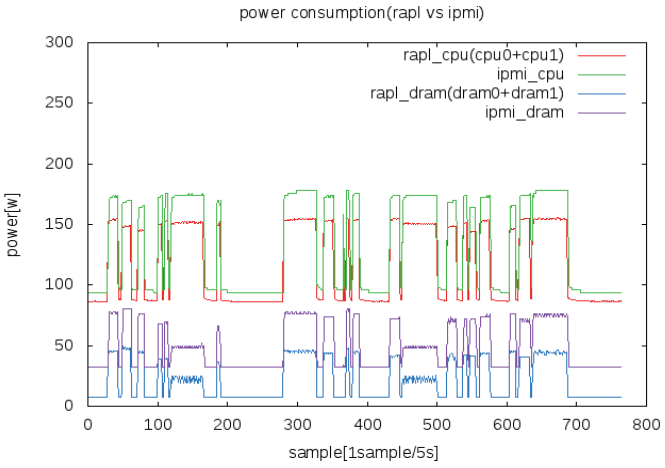

図 8 RAPL と IPMI の比較

5.3 RAPL による電力モニタと IPMI による電力モニタ

前節の評価では RAPL を用いて電力制約を与え，RAPL が提供する CPU の電力値をもとに結果を示している．2.5 節で述べたように RAPL は CPU のパフォーマンスカウンタを用いて CPU ソケット単位の消費電力を出力する．そのため，電力計での測定値との間に差があることも考えられる．そのため，RAPL の値と IPMI の値を比較した．これらは前節と同一の環境において，ジョブを実行している 1 ノードに着目している．これらを図 8 に示す．

これらの結果より IPMI の電力が総じて高い結果となった．DRAM については 30W 程度も IPMI が高い結果となり，CPU に関しては 25W 程度高い結果となった．IPMI による電力計測の場合，CPU やメモリ向けのコア電圧を生成するための DC-DC コンバータの電力を含んでいるためと考えられる．また，オフセットの違いはあるものの，RAPL と IPMI の電力値の挙動は似ており，オフセットに対する考慮は必要であるが，スケジューリング等の電力ヒント情報として RAPL の値をもとにすることは現実的であると言える．

また，RAPL のモニタリング値は CPU や DRAM のみをサポートしており，ファンやディスク，電源部の電圧変換ロス等の電力が含まれていない．この影響を調べるため，IPMI を用いて各コンポーネント（CPU，メモリ，ケースファン）やノード全体の電力を確認した．図 9 に電力値を示す．ipmi_fan はファン電力，ipmi_cpu は 2 ソケット分

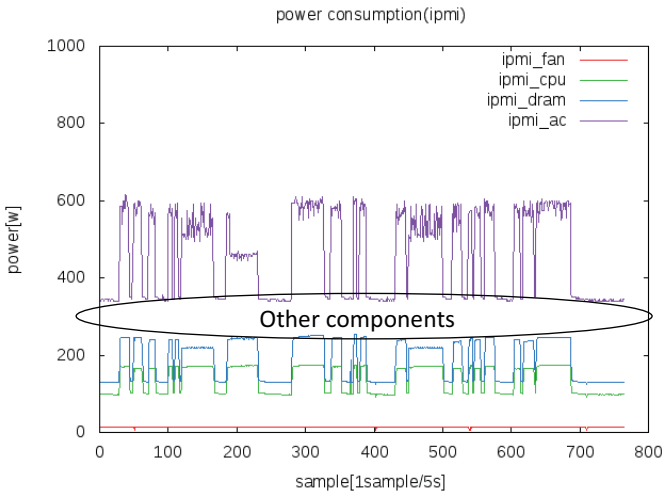

図 9 消費電力の内訳 (IPMI)

の CPU 電力，ipmi_dram は DRAM の電力値を示している．また，ipmi_ac は AC コンセント部で計測した計算機全体の消費電力を示している．この結果，1 台のノードは 300W 以上の電力を消費し，ジョブ実行中は 600W を超えることが分かった．また，CPU とメモリ以外の部分がノードの半分以上の電力を消費していることが確認された．また，ipmi_fan が示すケースファンは最大でも 15W 程度しか電力を消費しないことが分かった．

6. まとめ

本稿では，将来の電力制約適応型システムを見据えて，電力制約を超えるようなハードウェア資源が設置されたHPC システム環境を想定し，性能・電力消費特性について調査を行った．

オーバープロビジョニング環境において，各ジョブに対し厳しい電力キャップ値を与えることで，各ジョブの実行時間は伸びるものの，同時に多数のジョブを実行させることが可能となり，全体として効率的にシステムを利用できる可能性があることが分かった．

一方で，ジョブの特性がシステム全体の電力消費に与える影響や，どの程度のハードウェア資源を設置すべきかなどの設計方針はまだ明らかではないため，今後も調査が必要である．また，効率的な電力制約適応型システム向けスケジューリング手法を検討することも今後の課題である．

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．

参考文献

[1] TOP500 Supercomputer Sites http://top500.org/
[2] Power Management Framework for Post-Petascale Supercomputers http://www.hal.ipc.i.u-tokyo.ac.jp/research/pompp/
[3] Tapasya Patki, David K. Lowenthal, Anjana Sasidharan, Matthias Maiterth, Barry Rountree, Martin Schulz, Bronis R. de Supinski :Practical Resource Management in Power-Constrained, High Performance Computing, High Performance Parallel and Distributed Computing (HPDC) 2015.
[4] Thang Cao, Yuan He and Masaaki Kondo :Demand-Aware Power Management for Power-Constrained HPC Systems, Conference: The 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) 2016.
[5] Dror G. Feitelson and Ahuva Mu'alem Weil. :Utilization and Predictability in Scheduling the IBM SP2 with Backfilling, In Proceedings of the 1st Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing (IPPS/SPDP) 1998.
[6] Shoaib Kamil, John Shalf and Erich Strohmaier,: Power efficiency in high performance computing, IEEE International Symposium on Parallel and Distributed Processing (IPDPS), 2008.
[7] Laros, J.H., Pedretti, K.T., Kelly, S.M., Vandyke, J.P., Ferreira, K.B., Vaughan, C.T and Swan, M.: Topics on measuring real power usage on high performance computing platforms, IEEE International Conference on Cluster Computing and Workshops, 2009.
[8] Rong Ge, Xizhou Feng and Cameron, K.W. : Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters, Proceedings of the ACM/IEEE SC 2005 Conference Supercomputing.
[9] Yoshihiko Hotta, Mitsuhisa Sato, Hideaki Kimura, Satoshi Matsuoka, Taisuke Boku and Daisuke Takahashi, :Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster, International Parallel & Distributed Processing Symposium (IPDPS), 2006.
[10] Deva Bodas, Justin Song, Murali Rajappa and Andy Hoffman, :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) 2014.
[11] Efraim Rotem, Alon Naveh, Avinash Ananthakrishnan, Eliezer Weissmann and Doron Rajwan, :Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge, Micro, IEEE, Volume:32 , Issue: 2, pp.20-27(2012).
[12] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna and Christian Le :RAPL: Memory power estimation and capping, Proceedings of the 16th ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED), 2010.
[13] Simple Linux Utility for Resource Management http://slurm.schedmd.com/
[14] Motoyoshi Kurokawa, :The RICC log, http://www.cs.huji.ac.il/labs/parallel/workload/l_ricc/index.html/