Vol.2015-HPC-151 No.1
2015/9/30

電力制約を考慮した資源管理を行うリソースマネージャの実装と評価

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，電力制約を考慮しつつノードの資源管理やジョブスケジューリングを行うことや，システムの資源管理機構から電力ノブを制御することが必須となる．我々は，多くの HPC システムにおいて用いられている Slurm リソースマネージャに対して電力制約を考慮した資源管理手法を実装した．本稿では，その概要と，電力を考慮したジョブスケジューリングを用いた場合の性能評価について述べる．

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 Tianhe-2 は 33 ペタフロップスの性能を達成するために 18 MW 近い消費電力を必要としており，京は 10 ペタプロップスの性能を得るために 12 MW の消費電力を要する [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 20〜30MW 程度の電力で，現在の 30〜50 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．そのため，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，通信インテンシブなジョブは CPU やメモリの利用効率が低く，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．そこで，これらのジョブ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．そのためには，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・投入することが重要となる．

[†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

そこで，本稿では電力制約を考慮し資源管理を行うリソースマネージャの設計について報告する．さらに，提案する電力制約適応型リソースマネージャを HPC システムにおいて多く用いられる Slurm リソースマネージャ [3] を拡張する形で実装し電力の性能評価を行った．Slurm は Top500 システムのうち最も利用されているリソースマネージャである．

以降 2 章では資源管理手法について関連技術や研究を示し，電力制約適応型リソースマネージャの要件を明らかとする．これらの要件を受け，3 章では電力制約を考慮した資源管理手法の実装について示し，4 章で性能評価を示す．最後に 5 章にてまとめを述べる．

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，ジョブと計算ノードの対応を管理する．リソースマネージャには 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 [4] 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．

リソースマネージャの実装の 1 つである Slurm も同様にユーザーからのジョブ実行要求をジョブスケジューラが管理し，ジョブと計算ノードとの対応をノードスケジューラが管理している．これらの概要を図 1 に示す．ジョブスケジューラはユーザーから実行ファイル情報，利用ノード数，ランク数，実行期間，開始時刻などのジョブ情報を受け取る．これらジョブ情報はジョブキューに投入され，基本的に First Come First Served (FCFS) ポリシーで管理される．Slurm のジョブスケジューラではこのジョブキューに投入されたジョブ情報に対し，計算資源有効活用のためバックフィリングを行う．バックフィリングの際は定期的にジョブキューを探索し，ジョブが利用する計算ノード数や実行時間を考慮した上で，計算ノード全体の利用ノード数が最大になるようにジョブキュー内のジョブを並び替える．ジョブキューの中の特定のジョブを選択し，ノードスケジューラに対して選択したジョブが将来実行可能となる開始時刻を問い合わせる．この結果を受け，ジョブキュー内のジョブ情報の再スケジューリングを行う．

ノードスケジューラには 2 つの役割がある．1 つ目はジョブキューの先頭のジョブ情報を取り出し，計算ノードに割り当てる役割である．2 つ目はバックフィリングの問い合わせに対し，問い合わせのあったジョブが将来実行可能となる時刻を返答するものである．実際の計算ノードの割り当ては行わないものの，将来の計算ノードの利用状況を計算し，問い合わせのあったジョブの開始時刻を予測する．基本的にこれらの計算ノードの割当ては，計算を行わない計算ノード数が最小になるようにしている．このような従来の割り当て方法では，電力制約を考慮していないため，電力制約適応型システム向けにはいくつかの拡張が必要となる．

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 [5][6]．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の役割は重要である．リソースマネージャは電力制約適応型システムにおいて 2 つの役割を持ち，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を管理する．具体的にはジョブが消費する電力，電力制約，全計算ノードの利用状況を考慮したうえで，ジョブと計算ノードの割り当てを管理することにより，電力制約の中で利用できる消費電力が最大となるようにし，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重要である．具体的には，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する必要がある．個々のジョブが消費する電力のピークを制約しつつ，計算ノード全体として電力制約を超えないようにすることが重要である．

2.3 ジョブの電力特性

ジョブが計算ノードにて消費する電力は，多くの場合において設計最大消費電力を下回る．これは，ジョブ中の計算においてボトルネックとなる部分があり，そのボトルネックがジョブ全体の性能を制約し，十分にハードウェアの資源を利用できないためである．また，これらの電力特性はジョブの演算内容に大きく依存し，ジョブの特性でもある．

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

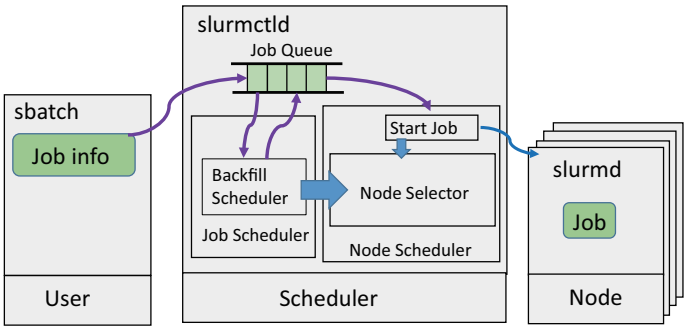

図 1　Slurm による資源管理

が重要となる．これらの電力特性をスケジューリングに用いることで，電力を考慮したリソース管理が可能となる．

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，各ジョブの必要消費電力を参考にスケジューリングを行うため，その最適な電力ヒントを見つけることが重要な役割を持つ．

これらの電力ヒントには下記の 3 つタイプが考えられる．

- 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，明示的に電力ヒントを与える．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を付加するものである．スケジューラは与えられた電力ヒントを参考にするため，スケジューリングを簡単化できる特徴がある．しかし，ユーザーがジョブの電力特性を理解することが必要となる．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あげられる．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，ジョブに対し厳しい電力制約を与える方式も考えられる．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アプローチである．各ジョブに対し，常に電力制約が働くような低い電力制約を与え，全体の電力制約の中で，消費電力が最大になるようにジョブを実行することができれば，電力を有効に利用でき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 [7] を行うものや，過去のジョブ実行をベースにプロファイリング [8] を行い電力特性を抽出する手法がある．コンパイラ等で自動的に解析を行えばユーザーが電力特性を考慮する必要はない．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などの情報より，動的に電力ノブの調整を行う手法 [9] などがある．プログラムの動的な挙動変化などに対応可能である特徴を持つが，数万台規模の非常に多くの計算ノードをリアルタイムに制御する必要があり，課題も多い．

3.　Slurm への電力を考慮した資源管理の実装

3 章の設計を元に Slurm リソースマネージャを拡張する形で，電力制約を考慮したリソースマネージャの実装を行った．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．

3.1　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，ユーザーがジョブの実行要求と合わせジョブの電力ヒントを与える実装とした．スケジューラはユーザーから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の追加，スケジューラの改良，電力モニタの追加，計算ノード上の電源管理デーモンの追加を行った．

3.2　電力ヒント部の実装

先にも述べたように，本稿は単純ではあるが電力ヒントとして，ユーザーがジョブが要求する最大消費電力を電力キャップ値として与えることとした．これにより，各ジョ

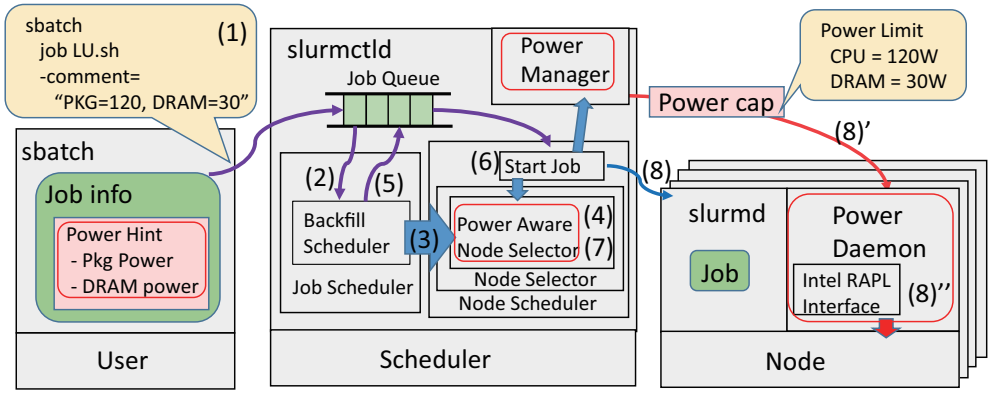

図 2　電力を考慮した slurm リソースマネージャ

ブが消費する電力を予測することが可能となり，計算ノード全体の電力制約を考慮したスケジューリングが可能となる．具体的には，ジョブに対し 1 つの CPU パッケージと DRAM が利用できる電力キャップ値を与えることとした．

3.3　ジョブの実行要求と電力制約の付加

Slurm ではユーザーはジョブ情報をジョブキューに投入することでジョブ実行をスケジューラに依頼する．Slurm ではこれらを sbatch コマンドを用いて行う．sbatch コマンドには実行するジョブの実行ファイル情報，利用ノード数，ランク数，実行期間，開始時刻等を合わせて下記のように指定する．

```
sbatch job.sh
```

job.sh にはジョブの実行ファイル情報，利用ノード数，ランク数を記載し，sbatch コマンドを用いてジョブキューに投入する．

これらの情報は Slurm のスケジューラである slurmctld がスケジューリングに用いる．そこで，本実装ではこの sbatch を行う際に，電力制約も与えるように拡張を行った．具体的には，sbatch 時にコメントとして CPU の電力制約，DRAM の電力制約を与えることができるように拡張を行った．具体的には

```
sbatch job.sh --comment=''PKG=120, DRAM=30''
```

に示すように sbatch と合わせジョブの情報と電力制約を指定する．PKG は 1 ソケットの CPU が利用可能な最大消費電力を示し，DRAM は 1 ソケットあたりに接続された DRAM に対する電力制限を示す．ジョブに対しコメントを与える機能は予め Slurm がサポートしているため，Slurm のインタフェースを変えることなく，最小限の拡張で電力制約を与えることが可能となる．

3.4　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のノードスケジューラにて行われるため，このノードスケジューラ内に拡張を行った (図 2 中央赤枠内)．ノードスケジューラではジョブが実行可能であるかを確認する．具体的には，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．そのため，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電力が残っているかを確認する機能を追加した．これらの拡張部分はジョブキュー内の最適化を行うバックフィリングとジョブを計算ノードに割り当てる際に用いられる．さらに，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に依頼する．また，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ようにパワーデーモン（図 2 右側）が導入されている．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．以下は，本リソースマネージャ内部の連携の詳細である．

(1) ジョブの投入と電力制約の付加（図 2(1)）
ユーザーはジョブの電力ヒントを，ジョブを投入する際に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) バックフィル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．さらに，これらを定期的に繰り返す．

(3) ジョブ開始予想時刻の問合せ（図 2(3)）

バックフィルによるジョブキューの最適化では，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，選択したジョブが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を行う．

(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．

3.5 計算ノードの電力制御

各計算ノードではスケジューラから与えられた電力制約を守り，ジョブ実行を行う必要がある．本実装では，CPU と DRAM に対して電力の制約を課すものとする．CPU との電力制約には Intel RAPL[10][11] を用いている．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，ソケット単位で消費する最大電力を設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電源制御を行う．また，ソケットに接続された DRAM に対しても，同様に電力制約を行うことが可能であり，本実装でもこれらを用いている．

4. 評価

本章では電力制約を考慮した Slurm リソースマネージャの機能確認と性能電力評価について示す．評価では電力制約を守りつつ，ジョブの実行が可能であることを実際の計算ノードを用いて確認する．また，電力制約の違いによる性能変化を確認するため，演算時間の変化や消費エネルギーの変化について確認する．

4.1 評価内容

本実験ではリソースマネージャの資源管理と電力結果についての評価を行う．そのため，計算ノード数を超える多数のジョブを同時に投入し，ジョブ実行中の計算ノードの電力を計測する．

評価に用いたジョブは NPB 中の CG, EP, MG の 3 つのカーネルとし，問題サイズは class C, class D を用いた．また，利用するノード数を変えたジョブを用意した．ジョブの電力制約は，一般的に利用可能な範囲内の制約値をランダムに与えた．さらに，これらの中からランダムに複数のジョブを取り出し，ジョブセットとして定義した．このジョブセットの中のジョブをシステムに投入し，計算ノード全体の電力制約を変えた場合について評価を行った．

本稿では以下の項目を評価する．

- 電力制約を満たすスケジューリングの実現：
  計算ノード全体の電力制約を超えることなく，ジョブのスケジューリングを行い，ジョブ実行ができることを実機の PC クラスタ環境にて確認する．そのため，電力制約を変えた場合について，制約電力の超過が起きないことを確認する．
- 実行時間・平均電力・消費エネルギーの変化：
  電力制約を課すことにより，同時に利用できる計算ノード数が減ることが予想される．これにより，全ジョブの実行に要する時間が変化する．そのため，全ジョブの実行時間や電力性能の変化について評価を行う．全ジョブの実行に要する時間，平均消費電力，エネルギーについて調査する．

4.2 評価環境

評価には表 1 に示す計算ノード 4 台を用いた．導入時期の違いにより，2 種の計算機が混在となっている．それぞれの計算ノードは，Xeon プロセッサを 2 基搭載する 2 ソケットマザーボードを持つ．計算ノード (Node[1-4]) とスケジューラノード (Scheduler) から構成され，ネットワークは一般的な 1Gbps の LAN を用いている．また，Slurm は最新安定版の 14.11.6 をベースに改良を行った．

電力の計測には RAPL に搭載されている電力計測機構を用いた．RAPL の電力計測機能は CPU に内蔵された機能であり，CPU パッケージごとに電力計測が可能である．ま

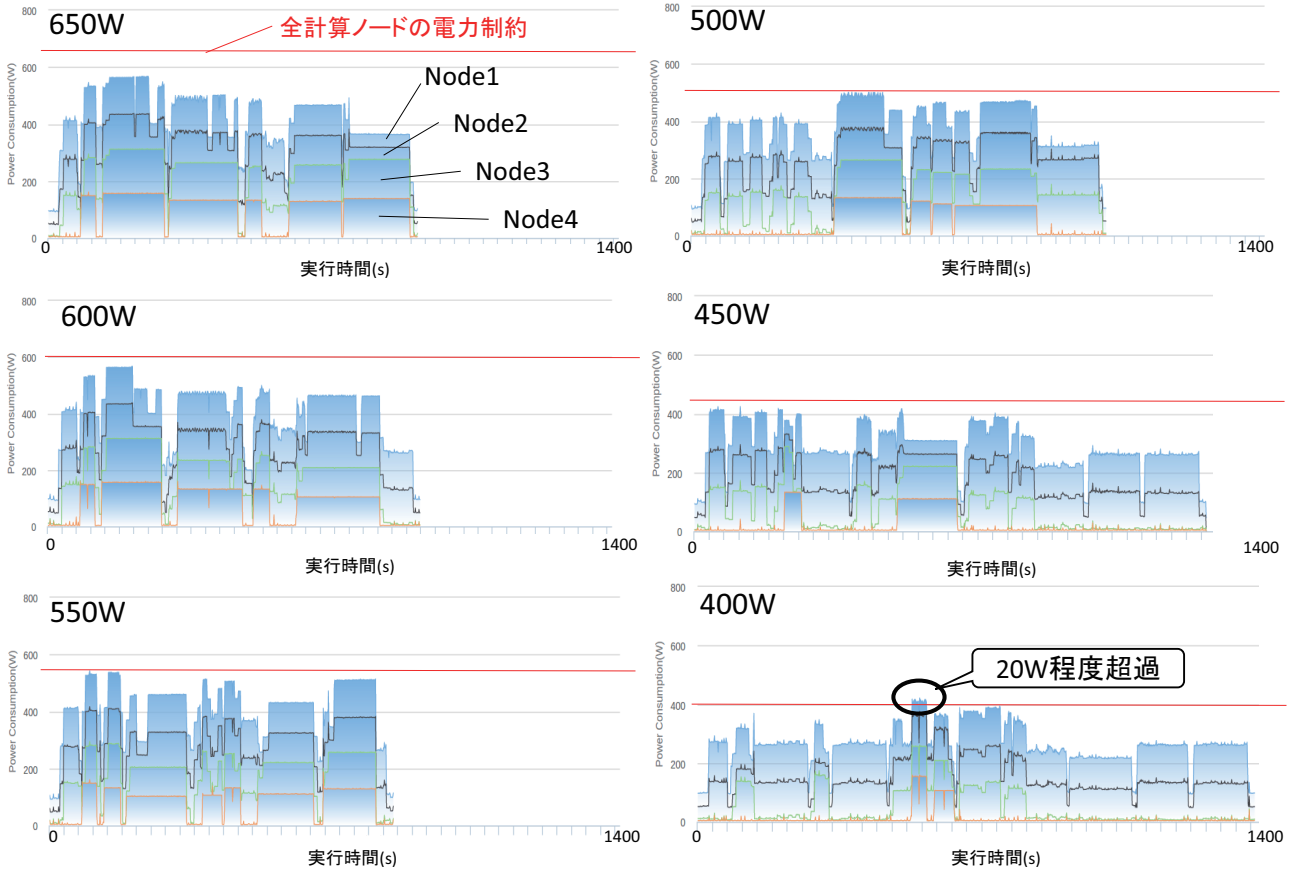

図 3　電力制約を変化させた場合のジョブ実行中の消費電力

表 1　計算ノード・管理ノードの仕様

Node[1-2]		
Processor	Intel Xeon E5-2670 v2	(2 sockets)
	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	8GB x 8
Node[3-4]		
Processor	Intel Xeon E5-2690	(2 sockets)
	Num. of Cores	8
	Frequency	2.9Ghz
	L1 Cache	32KB I + 32KB D per core
	L2 Cache	256KB per core
	L3 Cache	20MB per chip
DRAM	Size	8GB x 8
Scheduler		
Processor	Intel Core2 Duo E7200	
	Num. of Cores	2
	Frequency	2.5Ghz
	L1 Cache	32KB I + 32KB D per core
	L2 Cache	3MB per chip
DRAM	Size	2GB x 2

た，CPU ソケットに接続された DRAM の消費電力も同時に計測可能である．そこで，これらの CPU と DRAM の電力について計測を行った．RAPL による電力計測精度は比較的高いことが報告 [12] されており，評価として有効であると考えている．また，本評価では CPU と DRAM についての電力を示しており，他のストレージやディスク，ファンなどの電力は含まれていない．しかし，CPU と DRAM の消費電力は計算ノードの電力と強く相関があることが報告 [12] されているため，CPU と DRAM のみの計測でも，提案手法の有効性は示すことができると考える．また，電力計測は計算ノードのみを行い，スケジューラノードは含まない．

4.3　電力制約を変えた場合のスケジューリング結果と電力

本評価では，全計算ノードの電力制約が 650W, 600W, 550W, 500W, 450W, 400W とした場合の評価を行った．最大を 650W とした理由は，事前にベンチマークテストを行った結果をもとにしている．事前評価では 1 ソケット辺りの電力が，最大 80W 程度であった．本環境は 2 ソケット CPU からなる計算ノードが 4 台あるため，最大値を 650W とした．また，ジョブには 4.1 節で示したベンチマークから 20 個のジョブを行うようにしている．

4 台の計算ノードを用いて全計算ノードの電力制約を変えた場合のジョブ実行中の電力を図 3 に示す．この結果，400W を除く電力制約条件下において全計算ノードの電力

制約を守りつつジョブ実行が可能であることを確認した．電力制約を 400W とした場合において 20W 程度，電力制約を超過する場合があった．これは，RALP における電力制御がうまく働いていないためと考えられる．数秒程度の短時間に消費電力が大きく変わるジョブに対しては，RAPL による電力制約がうまく働かない．電力の超過ペナルティは計算機センターの利用ポリシーに大きく依存する部分があるが，電力超過を抑制する方法は今後の課題である．

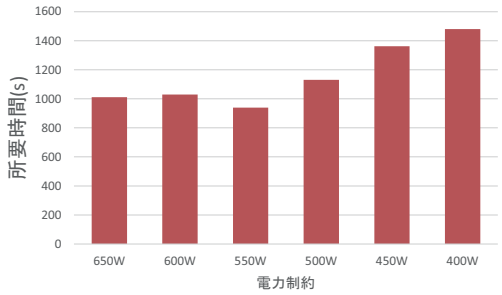

図 4 電力制約を変えた場合の全ジョブの実行に要した時間

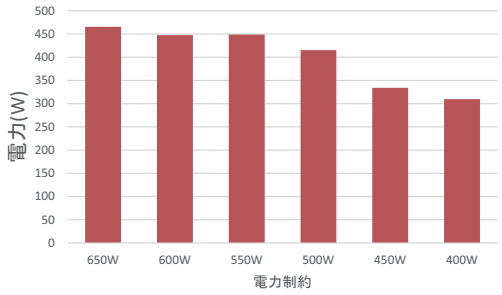

図 5 電力制約を変えた場合の平均実行消費電力

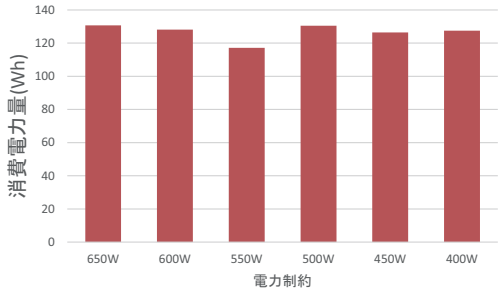

図 6 電力制約を変えた場合の消費電力量

4.4 電力制約下における実行時間・平均電力・エネルギーの変化

また，全ジョブを実行するのに要した総実行時間，平均消費電力，全エネルギーについて，それぞれの電力制約ごとに算出を行った．それぞれ，図 4，図 6，図 5 に示す．

全ジョブを実行するのに要した時間は，電力制約が厳しくなるにつれ，長くなることが分かった．これは，同時に実行可能なジョブ数が減ったためである．また，650W 時と 600W 時では実行時間に大きな差が見られなかった．これは，本実験では少ない計算ノードを用いて実験を行ったため，スケジューリング結果が大きく変わることがなかったためである．また，550W の際，実行時間が最も短くなっている．これは現在調査を行っているが，計算ノードの構成の差異によるものと考えている．ジョブと計算ノードの割り当てが変化したためと考えている．

平均消費電力は電力制約を厳しくするごとに，低下する傾向があることがわかる．これは電力制約により，同時に実行可能なジョブが減るためである．650W, 600W, 550W では大きく平均電力に差が見られない．これは，ジョブの実行に要する電力が計算ノード全体の電力制約より小さかったためである．

ジョブを実行する際のエネルギー消費電力量については，全体的に大きな変化は見られない結果となった．これは，制約が厳しくなるにつれ，実行時間が伸びるが，その分電力が減ったため，エネルギーとして大きな変化がなかったためと考えている．一方で 550W の際はエネルギーが削減されている．これは，前にも述べたように実行時間が変化したためである．ジョブに割り当てられる計算ノードの構成の差異により，実行時間が変化したためと考えている．

5. 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，HPC システムの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環境を想定し，計算ノード数やノード電力を考慮したリソースマネージャの初期実装について提案を行った．本実装ではジョブ単位で利用可能な電力値をヒントとしてリソースマネージャに与えることにより，HPC システム全体の電力制約を守るようにした．評価では，計算ノード全体の電力制約を大きく超えることなく，ジョブを実行することができた．一方で，利用した計算ノードの台数制約より，電力効率における考察を行うことができなかった．そのため，十分な計算ノードがある環境での評価が今後の課題である．また，現在はジョブの投入の際に与えられた電力ヒントのみを用いているため，ジョブ実行中に柔軟に対応することができない問題があり，動的な電源制御も課題である．

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] Simple Linux Utility for Resource Management http://slurm.schedmd.com/
[4] 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-98), pages 542-547, Los Alamitos
[5] Kamil, S., Shalf, J. and Strohmaier, E.: Power efficiency in high performance computing, IEEE International Symposium on Parallel and Distributed Processing, 2008. IPDPS.
[6] 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.
[7] 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,
[8] Hotta, Y., Sato, M., Kimura, H., Matsuoka, S., Boku, T. and Takahashi, D.: Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster, International Parallel & Distributed Processing Symposium , 2006. IPDPS.
[9] 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 Pages 21-30
[10] Rotem, E. , Naveh, A., Rajwan, D., Ananthakrishnan, A. and Weissmann, E., :Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge, Micro, IEEE, Volume:32 , Issue: 2, pp.20-27(2012).
[11] David, H., Gorbatov, E., Hanebutte, Ulf 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.
[12] カオ タン，和田 康孝，近藤 正章，本多 弘樹，RAPL インタフェースを用いた HPC システムの消費電力モデリングと電力評価，IPSJ 第 141 回 HPC 研究会，2013 年