バッチスケジューラ連動型省電力ツールの開発と運用

原田 浩[1]　藤本 大輔[2]　蛯原 純[2]　實本 英之[2]　平野 光敏[1]

概要：東京大学情報基盤センターでは、ジョブスケジューラと連動し、ジョブ投入可能な計算ノードだけを起動し、ジョブが投入される予定の無い計算ノードを停止する省電力ツールを開発した。開発した省電力ツールを東京大学が HPCI に計算機資源として供出している GPU クラスタの運用に使用した。HPCI の実運用下で 5 日間の消費電力を計測したところ 21.61 %の電力削減効果を得た。

1. はじめに

東京大学情報基盤センターでは、HPCI の計算資源として GPU クラスタ [1] を提供している。GPU クラスタにはいくつかの HPCI 課題が割り当てられているが、ユーザの利用率には変動があり、常に計算ノードにジョブが割り当てられているとは限らない。低使用率のクラスタ計算機では、ジョブを実行していないアイドル状態の計算ノードを停止することによって消費電力量の削減が期待できる。そこで、東京大学情報基盤センターでは、GPU クラスタがジョブスケジューラとして採用している GridEngine[2] と連動し、アイドル状態の計算ノードを自動停止し、ジョブが投入された時だけ必要な数の計算ノードを自動起動するツール（以下、省電力ツール）を開発し、消費電力量の削減を図ることとした。省電力ツールは、GridEngine 以外のスケジューラにも対応可能なように、GridEngine とは独立したプログラムとして実装することとした。省電力ツールは、”qstat”等 GridEngine のコマンドを実行することでキューや計算ノードの状態を得ることができる。一方計算ノードの停止・起動等の操作には IPMI を利用可能である。GridEngine や IPMI のコマンド実行が容易であること、プログラムの移植性・生産性を考慮して、省電力ツールはシェルスクリプトによる実装とした。

2. 省電力ツールの設計と実装

省電力ツールは、計算ノードをジョブ投入状況に応じて適切に起動・停止することによって電力消費量の削減を狙ったツールである。計算ノードの起動・停止等の操作は、ノード計算機の状態遷移ととらえることができる。省電力ツールの設計・実装にあたって、計算ノードの状態を、以下の４つと定義することとする。

- 停止
  電源断でプロセッサが動作していない状態。電源投入、IPMI のブートコマンド等でジョブ実行不可状態に遷移する。
- ジョブ実行不可
  電源が投入され計算機は動作しているが、ジョブスケジューラがジョブを投入できない状態。シャットダウン中の場合は停止状態に遷移する。ブート中の場合は、アイドル状態に遷移する。
- アイドル
  計算機が正常動作し、ジョブスケジューラがジョブ投入可能で、かつジョブが動作していない状態。計算ノードにジョブが投入されると、ジョブ実行状態に遷移する。シャットダウンコマンド等を実行することで、ジョブ実行不可状態に遷移する。
- ジョブ実行
  計算ノードがジョブを実行している状態。ジョブの実行が終了すると、アイドル状態に遷移する。

2.1 GridEngine による計算ノードの状態遷移

GridEngine だけで GPU クラスタを運用した場合のノード計算機の状態遷移を図 **1** に示す。計算ノードはシステム起動時のみ停止状態からジョブ実行不可状態をへてアイドル状態に遷移する。一旦アイドル状態に遷移した計算ノードは、 GridEngine によるジョブ投入時にジョブ実行状態に遷移し、ジョブの終了によって、アイドル状態へ遷移する。以後、計算ノードはシステム終了や障害が発生しない限りアイドル状態とジョブ実行状態を交互に繰り返す。

2.2 省電力ツールによる計算ノードの状態遷移

GridEngine に加え、省電力ツールによって GPU クラス

[1] 東京大学 情報システム部情報基盤課
[2] 東京大学 情報基盤センター

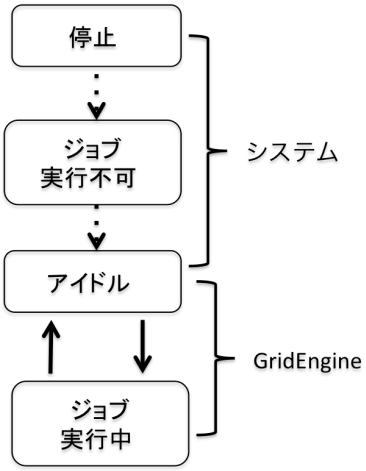

図 1　計算ノードの状態遷移：GridEngine

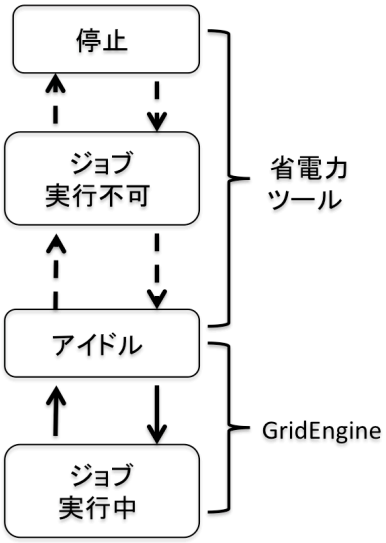

図 2　計算ノードの状態遷移：省電力ツール

タを運用した場合の状態遷移を図 **2** に示す。省電力ツールを用いた場合、計算ノードはジョブがキューに投入されない限り、停止状態を継続する。省電力ツールが実行待ちのジョブを検知し、ジョブ割当可能な計算ノードが停止状態の場合、省電力ツールは該当計算ノードを起動する。起動された計算ノードは、ジョブ実行不可状態を経由して、アイドル状態に遷移する。アイドル状態の計算ノードを検知した GridEngine は、ジョブを計算ノードに投入する。ジョブ投入された計算ノードは、ジョブ実行状態に遷移し、ジョブの終了とともにアイドル状態に遷移する。省電力ツールは、アイドル状態の計算ノードに割り当てるジョブが無いと判断すると、計算ノードを停止させる。停止させられた計算ノードは、ジョブ実行不可状態を経由して、停止状態に遷移する。

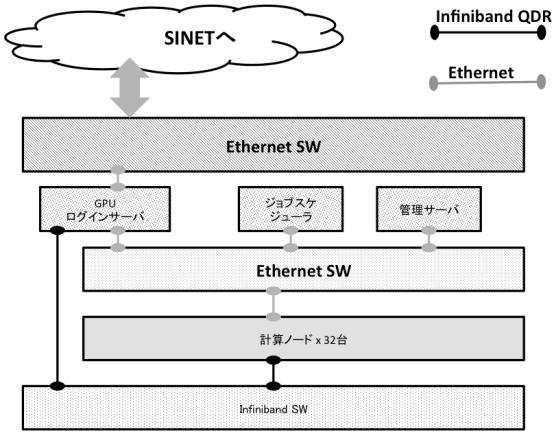

図 3　GPU クラスタ

2.3　省電力ツールによる状態遷移の実装

省電力ツールを実際に運用する GPU クラスタを図 **3** および表 **1** に示す。GPU クラスタはログインノード 1 台、ジョブ管理ノード 2 台、計算ノード 32 台から構成されるクラスタ計算機である。全てのノードはそれぞれ InfiniBand ネットワークで接続されている。OS として CentOS5.8 を、ジョブスケジューラとして GridEngine を採用している。ユーザは GridEngine を用いて計算ノードにジョブを投入・実行する。計算ノードは IPMI ポートを備えており、IPMI コマンドでの起動・停止等が可能である。

表 1　ハードウェア構成

サーバ	ログインノード 1 台 ジョブ管理ノード 2 台 計算ノード 32 台
CPU	Intel Xeon X5670 2.93GHz × 2 (2 ソケット 12 コア)
メモリ	48GB(計算ノード総量 1536GB)
GPU	Tesla M2050
NIC	InfiniBand 4x QDR

GPU クラスタ上での省電力ツールの実装について説明する。省電力ツールはポータビリティ、GridEngine 及び IPMI コマンドの利用を考慮し、シェルスクリプトで実装した。省電力ツールは cron によって 1 分ごとに起動され、必要に応じて計算ノードを停止状態またはアイドル状態へ遷移させる。省電力ツールの実装とは、図 2 に示した計算ノードの状態遷移の実装に他ならない。各状態遷移ごとに、GPU クラスタにおける省電力ツールの実装を説明する。

- 停止→ジョブ実行不可→アイドル
  省電力ツールは、GridEngine のコマンドを実行してキューに積まれたジョブと計算ノードの状態を調べる。実行待ちのジョブを検知し、ジョブ割当可能なノード計算機が停止状態なら、計算ノードを IPMI コマンドで起動する。起動された計算ノードはジョブ実行不可状態を経由して、アイドル状態に遷移する。起動後、15 分以上経過してもアイドル状態に遷移しな

い計算ノードは、何らかの障害が発生していると見なし、IPMI コマンドで強制再起動する。

- アイドル→ジョブ実行不可→停止
  省電力ツールは 5 分以上アイドル状態が継続している計算ノードを検知し、キューに割当可能なジョブが無いと判断した場合、計算ノードをシャットダウンする。シャットダウンされた計算ノードは、ジョブ実行不可状態を経由して停止状態に遷移する。

GPU クラスタではデバッグ等の用途のために、計算ノード 4 台はアイドル状態から停止させず、常時稼働させる。常時稼働の計算ノード 4 台は図 1 と同様の状態遷移となる。

3. 消費電力測定

GPU クラスタに省電力ツールを適用し、2013 年 10 月 21 日 00:00:00 から 2013 年 10 月 25 日 23:59:59 の 5 日間、計算ノードの消費電力を 1 分おきに計測した。計算ノードの消費電力は HP 社 iLO インタフェイスを用いて測定した。計算ノードの状態ごとの消費電力は**表 2** の通りである。

表 2 ノード計算機の消費電力

停止	0W
ジョブ実行不可	約 200W
アイドル	約 200W
ジョブ実行	240〜450W

4. 評価

各計算ノードの消費電力の測定結果から消費電力量を求めたところ、測定期間中の全計算ノードの総消費電力量は 882.507KWh であった。各計算ノードの消費電力の測定結果から、消費電力が 1W 以上の計算ノードを稼働ノードと見なした場合の稼働率分布を**図 4** に示す。横軸は時間、縦軸は各計算ノードとした稼働率である。計算ノードの稼働率は 68.32 %であった。ジョブの実行履歴から同様に横軸を時間、縦軸を各計算ノードとした利用率の分布図を**図 5** に示す。利用率は 56.47 %である。

測定からアイドル状態の計算ノード 1 台の消費電力は約 200W である。稼働率から停止状態の計算ノードのノード時間積を求めると 1214.208 ノード時間である。この値にアイドル状態の消費電力を掛けることで、ツールによる電力削減量は 242.84KWh であることがわかった。測定期間中の計算ノードの総消費電力量が 882.507KWh であるから、消費電力の削減率は 21.61 %である。測定結果を**表 3** にまとめておく。

計算機運用では、さらに冷却・空調の電力削減効果を見込めるので、実際の電力削減効果は 242.84KWh 以上であると考えられる。

稼働率を利用率に近づけることができれば更なる電力削減が期待できる。本測定では稼働率と利用率の差は 11.85 %である。主に以下の理由によるものであると考えられる。

- 常時稼働している計算ノードの存在
  計算ノード全 32 台中 4 台を即座にジョブを実行できるように常時稼働していることにより、最大で 12.5 %の差が生じる。
- 計算ノードのジョブ実行不可状態のノード時間積
  計算ノードの起動を開始してから GridEngine プロセスが立ち上がりジョブを実行できるようになるまでの間、また、ジョブの実行が完了しアイドル状態となってから計算ノードを停止するまでの間は、ジョブが実行されていないため、稼働率にのみ計上され利用率には計上されない。GPU クラスタで使用している計算ノードでは、1 度のジョブ実行で合算して 10 分程度がジョブ実行不可状態に費やされる。

表 3 測定結果のまとめ

稼働率	68.32 %
利用率	56.47 %
計算ノードの消費電力量	882.507KWh
計算ノードの削減電力量	243.30KWh
計算ノードの電力削減効果	21.61 %

5. まとめと課題

アイドル状態の計算ノードを自動停止する省電力ツールを開発し、GPU クラスタの実運用に用いた。利用率 50 %程の実運用下で、計算ノードの電力使用量を 243KWh（21.61 %）削減することができた。計算ノードの起動・シャットダウン時間を最適化することによりさらなる電力削減効果が期待できる。また計算ノードだけではなく、ネットワーク装置、空調・冷却装置についても計算機の稼動状態・負荷に最適化させる事などで更なる電力削減の可能性がある。ジョブスケジューラと連動することにより計算機資源の更なる消費電力削減の可能性を探っていきたい。

謝辞 本取り組みは文部科学省「HPCI の運用」の支援を受けている。

参考文献

[1] 東京大学 情報基盤センター, HPCI 関連サービス, http://www.cc.u-tokyo.ac.jp/other/hpci/.
[2] Grid Engine, http://gridscheduler.sourceforge.net.

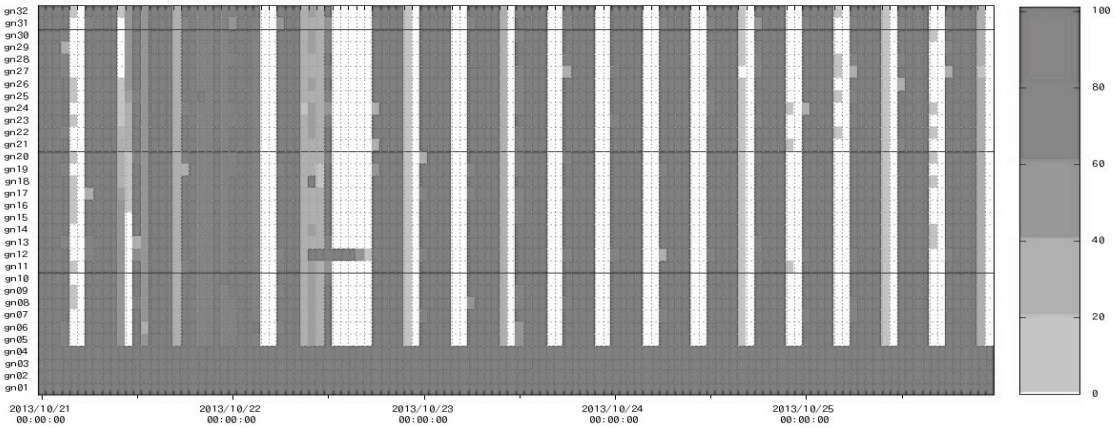

図 4 稼働率の分布

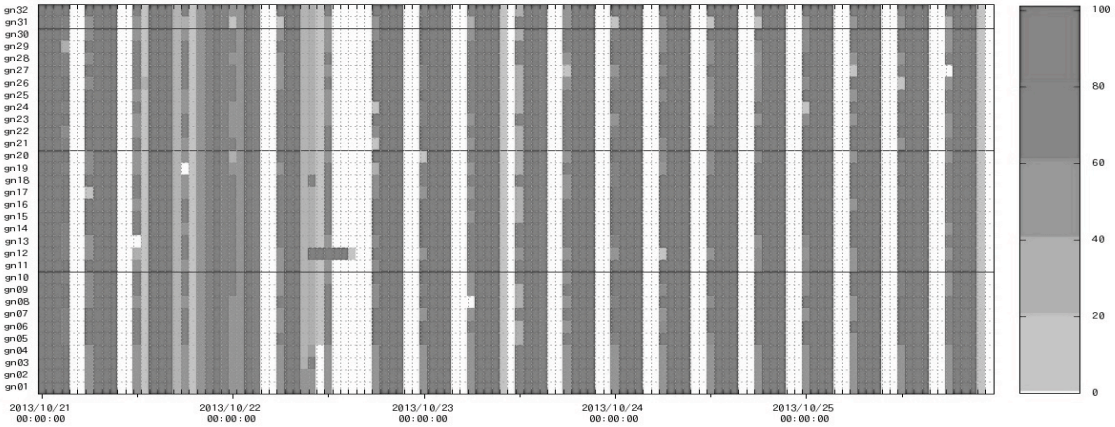

図 5 利用率の分布