3A-1

計算センタ高稼働率とジョブ待機時間短縮を実現するジョブスケジューラ

宇治橋　善史†　　久門　耕一†

（株）富士通研究所‡

1. はじめに

複数のクラスタシステムなどを持つ大規模計算環境においては、多種多様な規模・属性を持つジョブを効率的かつ平等に実行するために、ジョブ管理システムを利用する。多くの従来のジョブ管理システムではジョブキューによる管理方式を採用している[1]。このようなジョブ管理システムにおいて多種多様な規模・属性のジョブを管理するには、複数のジョブキューを作成し、各キューの優先度・ジョブ同時実行数制限などのチューニングパラメータの設定を行う必要がある。しかし、この設定値が不適切であると、資源利用の非効率、ユーザ間の不平等が発生する。そして、大規模な計算環境になると、キュー数が増加し、管理者のキュー設計困難化及び管理負担増加により、適切な運用が困難になる。

また、LSF[2]などのジョブ管理システムは、バックフィルスケジューリングポリシーを選択することが可能[1]で、FCFS(First Come First Served)スケジューリングと比較して高効率なリソース利用を実現しているが、ライセンス管理、複合クラスタ、ユーザ毎の実行数制限などとバックフィルを連携する柔軟なリソース管理はできない。

そこで、我々はこれらの問題を解決するため後述するメタジョブスケジューラの研究開発を行った。次章以降で、メタジョブスケジューラの特徴と、実績ログを元にシミュレーション実施した評価結果を述べる。

2. メタジョブスケジューラ設計思想

2.1 ジョブプール

ジョブキュー管理の欠点は、ジョブが、規模・属性によってキューに固定的に振り分けられ、状況に応じた柔軟な資源割り当てが不可能なことにある。そこで、我々は、ジョブを複数のキューで受け付けるのではなく、全ジョブを単一キューで受け付けてジョブプールで待機させ、各ジョブの規模・属性に従って、最適な計算資源を選択、割り付ける方針を採用した。

2.2 リソース記述式

メタジョブスケジューラは、計算資源などのリソース量を内部で保有し、ジョブ実行・終了時に、リソース量を増減することで、利用リソース・空きリソースを管理する。そして、ライセンスのような物理的実体がない対象をリソースにすることが可能であり、システム全体で制限値をもつ対象を、柔軟に管理することが可能である。

各ジョブが利用するリソース条件は、様々なものが考えられ、ライセンスリソースを利用して実行させたい場合、２つのクラスタリソースのどちらかで実行させたい場合、さらに複雑なものなど、様々な条件を指定する必要がある。そこで、メタジョブスケジューラでは次に記載するような、リソース記述式を導入した。

例えば、クラスタＡ(Clust_A)とクラスタＢ(Clust_B)のいずれか一方で、10 ノード利用して実行し、かつ、ライセンス(Lic)を 5 消費するジョブは、以下で記述できる。

```
(Clust_A : 10 | Clust_B : 10) & Lic : 5
```

メタジョブスケジューラは各ジョブのリソース記述式を解析し、その条件に従って後述するリソース予約表へジョブ割り当てを行う。

2.3 リソース予約表

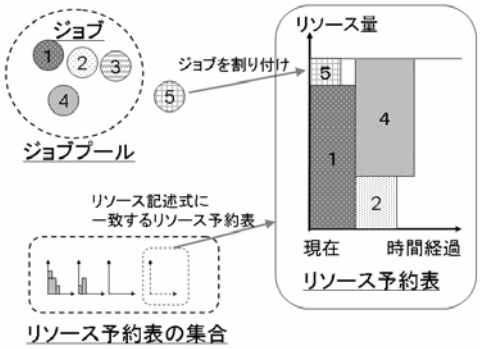

図 1　ジョブプールとリソース予約表概念図

メタジョブスケジューラは、リソースを時系列で管理するリソース予約表を内部で保有している。

スケジューリング処理は、以下の処理をジョブプールのジョブに対して先頭から順番に行うことで実行する。

1) ジョブプールから次の順番のジョブを選択。
2) ジョブが持つリソース記述式を解析。
3) リソース記述式の条件に一致する条件で、ジョブを割り付け可能なリソース予約表の場所を探索し、最も早く割り付け可能な場所に、ジョブを割り付ける(図１)。ジョブの利用リソース量と実行時間（予想実行時間）はジョブ投入時に指定される。
4) ジョブを割り当てた結果、割り当て時刻が現在時刻であれば、ジョブを実行する。割り当て時刻が現在でなければ、1)から処理を繰り返す。

スケジューリング処理は、ジョブ投入・終了などの契機で行い、その度にリソース予約表をクリアして先頭の

Job scheduler that improves usage efficiency and reduces waiting time of jobs

† Ujibashi Yoshifumi, Kumon Kouichi

‡ Fujitsu Laboratories Ltd.

ジョブからスケジュール処理を行う。
メタジョブスケジューラでは、このように、リソース記述式を導入することで、複数クラスタでのジョブの最適配置、ライセンス数などを統一的かつ柔軟に管理し、また、バックフィルが可能なリソース予約表で効率的なジョブ割り当てを実現した。

3. 導入効果 （シミュレーション）

メタジョブスケジューラの導入効果を検証するために、独立行政法人理化学研究所 RSCC(RIKEN Super Combined Cluster System)におけるメタジョブスケジューラ導入前の NQS(Network Queuing System)運用実績ログを元に、メタジョブスケジューラシミュレータでシミュレートし、導入効果を確認した。
メタジョブスケジューラシミュレータは、実績ログのジョブ投入時刻とジョブ属性を入力とし、スケジューリング結果として、ジョブ開始時刻と終了時刻を出力する。シミュレーションの対象期間は、3 日間で 7000 本のライフサイエンス系のジョブが大量に投入され、それらのジョブに邪魔をされ、長時間待機させられるジョブが発生した、2006/11/15～11/18 の期間とした。

3.1 計算資源利用率

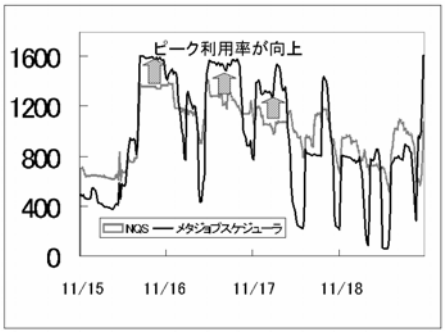

図 2 計算資源利用率（CPU 数）

RSCC 全クラスタ利用率の比較を図２に記載する。約 1800CPU の全計算資源のうち、NQS の実績では、1400 程度のピーク資源占有であったが、メタジョブスケジューラ導入により、1600 程度にまで向上する結果になった。実績ログのジョブ投入速度が遅く、残りの約 200CPU を活用しきれていないが、仮にユーザがジョブをより速く投入すれば、残り資源も利用可能と考えられる。

3.2 ジョブスループット

次に 7000 本のライフサイエンス系ジョブの投入・終了累積数の比較を図３に記載する。NQS 実績では、ユーザが 7000 本ジョブを投入開始してから、それらのジョブが全て終了するまでに約 52 時間かかっていることがわかる。それに対して、メタジョブスケジューラでは、7000 本ジョブ全てがジョブ投入とほぼ同時に実行された。結果として 31 時間で全ジョブ実行が完了したが、前記したように計算資源が余る状況なので、ジョブ投入速度がより速ければ、より早く完了するはずである。

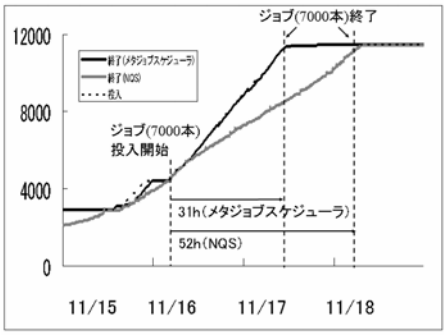

図 3 ジョブ投入・終了の累積数

3.3 待ちジョブ数

待ちジョブ数の比較を図 4 に記載する。NQS 実績と比較して、メタジョブスケジューラでは大幅に待ちジョブ数が減少している。高利用効率により先行ジョブが次々と完了し、後続ジョブの待ち時間減少を可能とした。

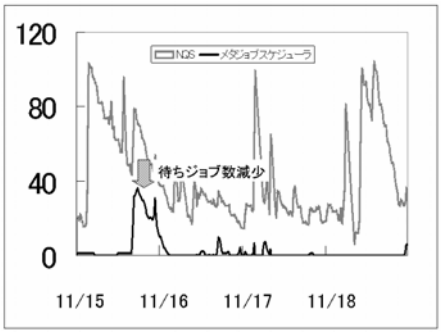

図 4 待ちジョブ数

4. まとめ

以上示したように、従来のジョブキューベーススケジューラより、大幅に高効率なジョブスケジューリングを実現することができることがわかった。また、このメタジョブスケジューラは RSCC に 2006 年 8 月から適用、さらに RSCC の次期システムである RICC(RIKEN Integrated Cluster of Clusters [3])にも本スケジューラを採用し、高い効率を実環境にて達成している。

謝辞

独立行政法人理化学研究所情報基盤センタ黒川様、重谷様に、メタジョブスケジューラを共同開発させていただき、シミュレーションのための貴重なジョブ履歴をご提供いただいたことを深謝申し上げます。そして、富士通（株）計算科学ソリューション統括部諸氏の多大なるご協力に感謝申し上げます。

参考文献

[1] Y. Etsion and D. Tsafrir, “A Short Survey of Commercial Cluster Batch Schedulers”. Tech. Report 2005-13, The Hebrew University, May 2005.
[2]http://www.platform.com/workload-management/high-performance-computing
[3] http://accc.riken.jp/ricc.html