ワークフローアプリケーションに対する計算資源割り当ての最適化

斎藤貴文 † 千葉立寛†,☆
佐藤仁 † 松岡聡†,††

1. 研究背景

近年の科学計算技術の発達に伴い，科学データは増加の一途を辿っている．このような大規模データに対して効率的にデータ処理を行うことが近年重要になってきている．

現在，様々な大規模データに対して並列分散処理を可能にするプログラミングモデルが存在するが，中でも複雑かつ自由に処理タスクを連携させてデータ処理を次々に適用し結果を導くワークフローシステムに注目が集まっている．ワークフローは既存のプログラムを組み合わせて構成しているのでプラットフォームの移植性が高く，また複雑なデータ転送や同期がユーザから隠蔽されるので，容易な分散並列処理を可能にする．

ワークフローはワークフローの形状や実行環境の性能に影響されるので，どの程度のリソースを用意すればワークフローが最適化されるかを知ることは難しい．また，タスクによって処理が異なるので，それに応じて最適な資源を割り当てることも考えなければならない．そのような性能最適化を行うためにはワークフローに適用可能な汎用的ワークフロー性能モデルが必要である．

本稿ではワークフローアプリケーションをタスクごとの性能モデルを構築し，小規模なワークフローをそれぞれの環境でテストして実行ログを習得し，そのデータからタスク毎の計算性能やI/O性能を見積もり，大規模データを扱うワークフローの性能を予測する手法を提案した．天文データ解析ワークフローのMontage[2]を対象としてTSUBAME2.0[3]及びAmazon EC2/S3[1]上で性能予測を行い，実性能との比較を行った．その結果．ある程度の性能予測は可能だったがモデル化されていないパラメータが性能に大きな影響を与えていることがわかった．

† 東京工業大学
☆ 現在，日本アイ・ビー・エム（株） 東京基礎研究所
†† 国立情報学研究所

2. 提案手法

ワークフロー W の実行時間 T_{total} は n 個のタスクの実行時間 $t_i(1 \leq i \leq n)$ の和で求めることができる．

$$T = \sum_{i=1}^{n} t_i$$

またタスクの実行時間は入力ファイルを読み込む時間 T_{read}，処理を行う時間 T_{calc}．出力ファイルに書きこむ時間 T_{write} の3つに分けることができる．

$$\begin{aligned} t_i = & T_{read}(S_{in}, J, N, P) \\ & + T_{calc}(S_{in}, J, N, P) \\ & + T_{write}(S_{out}, J, N, P) \end{aligned} \quad (1)$$

S_{in}，S_{out} は入出力ファイルのサイズ，J はタスクの総ジョブ数，N は処理を行うノード数，P は1ノードあたりのプロセス数で表す．

また個々のタスクをN-Nタイプ，N-1タイプ，1-Nタイプ，N-Nタイプの4つのタイプに分割する．このタスクのタイプによってファイルアクセスの挙動が異なると考えられる．

次に $Read$，$Write$，$Calc$ に費やす時間のモデル化を行う．ファイルのRead/Writeのスループットを B_{read}，B_{write} とするとき $T_{read|write}$ は以下のように表せる．

$$T_{read}(N, P, S_{in}, J) = \frac{S_{in}}{B_{read}} \times \frac{J}{N \times P} \quad (2)$$

$$T_{write}(N, P, S_{out}, J) = \frac{S_{out}}{B_{write}} \times \frac{J}{N \times P} \quad (3)$$

また，$Calc$ 時間は以下のようにモデル化する．

$$T_{calc}(N, P, S_{in}, J) = \nu_i \times \frac{J}{N \times P} \quad (4)$$

ν_i は1つのジョブを実行するのに必要な実行時間を表す係数である．

構築した性能モデルを検証するために TSUBAME2.0 で Montage を動かした．このモデルを用いた性能予測を行う手法を説明する．巨大なデータセットをもつワークフロー W_{large} に対して，サイズの小

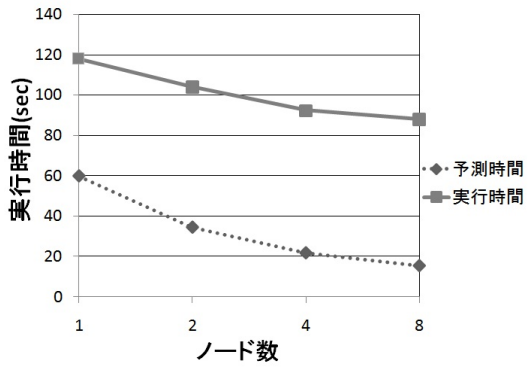

図 1　TSUBAME2.0 における予測値と実測値

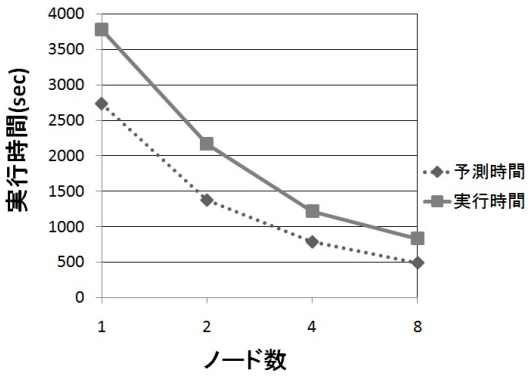

図 2　Amazon EC2/S3 における予測値と実測値

さいデータセットをもつワークフロー W_{small} を用意しこのワークフローの実行時に得られたログと入力データサイズからモデル式の係数を求め，性能予測を行う．

3. 評　　価

図 1 は TSUBAME2.0 における実行時間と予測時間を比較したグラフである．これより TSUBAME2.0 における実測はノード数の増加に対する性能向上が少なく，実測値と予測値の間には 2 倍以上の差が生じていることがわかる．

図 2 は Amazon EC2/S3 における実行時間と予測時間を比較したグラフである．Amazon EC2/S3 では性能向上が見られ， TSUBAME2.0 よりも予測の誤差は小さかったが， それでも実測値と予測値に大きな差がでる結果となった．

4. 考　　察

予測値と実測値の間に差が生じる原因を調べるため，実行ログの解析を行った．その結果モデル化されていないパラメータの影響によって二つの値に差が生じたことが確認された．

モデル化されたいないパラメータの 1 つはメタデータコストである．メタデータコストはファイルシステムにアクセスする際に生じる．実行ログの解析から，メタデータへのアクセス時間が I/O に及ぼす影響は無視できないことがわかった．

もう 1 つのパラメータはメモリキャッシュのヒット率である．中間ファイルの出力とその中間ファイルを必要とするジョブの処理が同一ノードで行われる場合，そのファイルがメモリキャッシュに乗っている可能性が高い．またノード数が増えるに従ってメモリキャッシュヒット率は低下する傾向がみられた．これはあるノードで実行されたジョブに必要な中間ファイルが他のジョブのキャッシュに乗ってキャッシュを利用できないことに起因する．このメモリキャッシュの影響を考慮していなかったので，正確な Read のスループットを見積もれていないことがわかった．Amazon EC2/S3 では 1 ノードあたりメモリが小さいのに対し，TSUBAME2.0 はメモリのサイズが大きく，メモリキャッシュの影響が大きかったため実測と予測で大きな差が生じたと考えられる．

5. まとめと今後の課題

本研究ではワークフローのタスクの特徴を分類してそれぞれについてモデルを構築し，性能予測する手法を提案した．また異なる計算環境でワークフローを実行しその結果の考察を行った．その結果，計算時間に関して実測と予測の差は小さかったが， I/O 時間には大きな差があらわれた．

今後の課題としては，ノードのメモリ容量やメタデータアクセスを考慮した性能モデルの改良や他のワークフローアプリケーションへの性能モデルの適用が挙げられる．

参　考　文　献

1) Amazon Web Services. http://aws.amazon.com/.
2) G. B. Berriman, A. C. Laity, J. C. Good, J. C. Jacob, D. S. Katz, E. Deelman, G. Singh, M. H. Su, and T. A. Prince. Montage: The architecture and scientific applications of a national virtual observatory service for computing astronomical image mosaics. In *Proceedings of Earth Sciences Technology Conference*, 2006.
3) TSUBAME2.0. http://tsubame.gsic.titech.ac.jp.