Vol.2011-HPC-130 No.3
2011/7/27

PARP: プロファイル比較に基づく並列アプリケーションの性能解析

加辺友也[†1]　田浦健次朗[†1]

本研究では並列化されたワークフローアプリケーションの性能を解析し，最適化を補助するためのツールとして，性能情報を関係データベースに格納し，異なる環境同士のそれを比較することによって解析を行う PARP for Workflow を提案した．比較に基づく既存の性能解析は MPI などに代表される SPMD 型の並列アプリケーションを対象にしていた．しかし，既存のデータベースではワークフローアプリケーションの性能情報を格納するのは困難である．本研究ではワークフローアプリケーションの性能情報を格納するためのデータベースを設計し，それを用いて異なる環境同士の性能を比較するための Web インタフェースのプロトタイプを実装した．評価においては実ワークフローアプリケーションを用い，性能比較を行うことで性能差の原因がどこにあるのか，部分的にではあるが解析可能であることを示した．

PARP: Relative Performance Analysis for Parallel Application

Tomoya Kabe[†1] and Kenjiro Taura[†1]

We proposed PARP for Workflow, a relative performance analysis tool of different conditions for parallelized workflow applications. It stores performance information in the relational database and it is aimed to support workflow developers or workflow engines' developers to optimize the workflow applications. Previous performance analysis based on comparison targeted at SPMD-type parallel applications like MPI. Existing performance databases, however, are unsuitable to store performance information of workflow applications. We designed a database schema to store performance information from workflow applications and implemented a prototype of Web interface for relative analysis between different execution conditions. In the evaluation we showed the result of a real-world workflow application and the possibility to partially analyze the cause of performance difference by comparison.

1. はじめに

広く研究され， MPI などで記述された SPMD(Single Program Multiple Data) 型のアプリケーションの並列化に対し，並列実行可能な実世界アプリケーションとしてワークフローアプリケーションがある．ワークフローアプリケーションは様々な種類のアプリケーションの組み合わせによって全体として 1 つの処理を行うタイプのアプリケーションである．処理は数分から数時間に及ぶようなものであり，より高速に処理が完了することが望ましく，現在では相互に依存しないジョブ間は同時実行可能であることに着目し，アプリケーションの並列化が進められている．そのような並列化されたワークフローアプリケーションを動作させるにあたって，動作させる環境が異なることで予想外の性能を示す場合は多々ある．これまでにある 1 回の実行における動作を解析する手法は提案されてきているが，異なる 2 つの環境での実行の結果，性能が大きく異なる際にその実行時情報を比較し，どのアプリケーションが性能差に大きく寄与しているかを解析するのに役立つ解析手法は示されていない．しかし，既に片方がよりよい実行性能を示していることを踏まえれば，性能の違いを解析することにより最適化を図るのはより合理的であるといえる．

一方，SPMD 型の並列アプリケーションに対して，実行時の性能を比較する手法は研究が進められてきた．本研究では SPMD 型並列アプリケーションでの性能比較手法を踏まえ，並列ワークフローアプリケーションの解析との違いを考察し，性能を比較するツールのプロトタイプ実装について述べる．

1.1 本稿の構成

2 節では対象とするワークフローの定義，その実行に関連する研究および，性能を比較することにより解析する既存の研究について述べる．3 節では本研究におけるプロトタイプ実装 PARP for Workflow について，設計方針および実装を述べる．4 節では評価として実世界ワークフローアプリケーション Montage を用いた解析例を示し，5 節でまとめと今後の課題について述べる．

†1 東京大学大学院 情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

Vol.2011-HPC-130 No.3
2011/7/27

2. 関 連 研 究

2.1 ワークフローアプリケーション

ワークフローアプリケーションはいくつかのアプリケーションの組み合わせにより全体として 1 つの動作をするアプリケーションであり，ここでは個々のアプリケーション間のデータのやりとりをファイルを用いて行うものとする．入出力のファイルはそれぞれ複数でありうる．アプリケーション 1 つ 1 つの実行をジョブ（タスクとも呼ばれる）と呼ぶことにする．すなわち，あるアプリケーションが実行開始するのは必要な入力が全て揃った後である必要がある．したがって，ジョブ間には入出力ファイルに関する依存関係が存在するものがある．互いに依存しないジョブも存在し，それらは資源があれば並列実行が可能である．

ワークフローアプリケーションは内部のアプリケーションをノードとし，入出力ファイルの依存関係をアークとする DAG (Directed Acyclic Graph) として，あるいはペトリネット[11] によりモデル化することができる[9),13)]．すなわち，入出力ファイルをトークンとし，システムの（特に入出力ファイルの）状態をプレース，そしてアプリケーションがトランジションとして記述される．この対応と，依存関係に基づいた並列実行可能性を**図 1** に簡略化した図として示す.

ワークフローは処理の流れであり，記述されたペトリネットを駆動するものはワークフローエンジンと呼ばれる．ワークフロー中で行われる各ジョブの実行時間はジョブ毎に異なるため，ワークフローエンジンは計算資源とジョブを動的に結びつけ，スケジューリングを行っていくことで実行時間の短縮を図っている．本研究ではワークフローエンジンとして GXP Make を用いる．これについて次節で述べる．

2.2 GXP Make

GXP Make[19] はワークフロー中のジョブ同士の依存関係を GNU Make の受理するスクリプト (Makefile として知られるもの) として記述し，それを並列に実行するためのワーフローエンジンの 1 つである．GXP Make では NFS や Lustre などによりファイルシステムがノード間で共有されていることを仮定し，GXP の提供する遠隔シェルを利用してジョブをリモート実行する．GXP Make は多くの UNIX 環境で利用できる Python や GNU Make をベースに動作し，展開コスト，学習コストともに低い一方，強力な記述力により複雑なワークフローも記述することができる．GXP Make は実行した各ジョブに関する性能情報を出力（現在は CSV 形式）しているため，今回はこの情報を用いて性能解析を行うこととした．

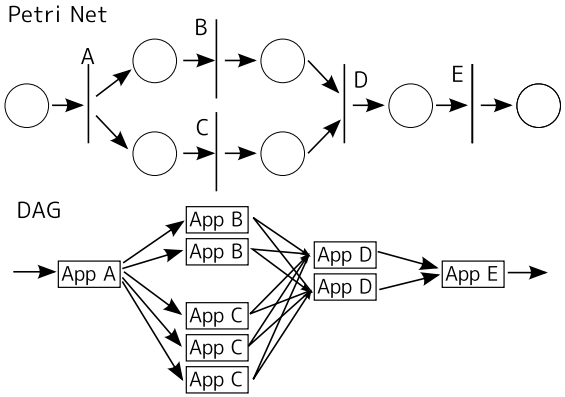

図 1: ペトリネットと DAG の対応
Fig. 1: Correspondence between Petri Net and DAG

2.3 ワークフローアプリケーションの性能解析

ワークフローの概念は古くから存在しているが，計算資源が豊富になり，並列化して性能を向上させる研究が活発になったのは比較的新しい話である．性能を評価するための計量に関する研究もまだ活発に行われている[12),18),20)]．また，10) では，ジョブが実行される場所と入力ファイルの存在場所が異なっていればそれを転送しなければジョブが処理できないが，この転送時間を考慮したスケジューリング戦略の性能をシミュレーションにより評価している． 21) はワークフロー実行中に逐一性能情報を記録していき，障害発生時の復旧やオンラインでの性能モニタリングなどを可能にした場合の性能評価を行っている．

2.4 性能の比較解析

2 つの異なる環境における並列アプリケーションの実行時情報を比較し，最適化に用いるための研究は多くなされている[3),4),6),8),14),17)]．中でも主流は実行時の性能情報を関係データベースに格納しておき，必要なときに任意のアプリケーションを用いてデータを解析できるようにしておくというものである．そのため，データベースを用いている研究[4),5),8)] においては SPMD 型の並列アプリケーションを念頭におきつつ，可能な限り一般的なデータベースを設計している．3) ではデータベースを用いるのではなく，性能ログは取得したところにそのままの形式で保管しておき，利用したいときに統一されたビューを提供する手法を提案しているが，大容量のデータを解析時に高速に処理する目的を考えると，事前にデータ

ベースに格納しておく方がよいと考えられる．

これらのツールは SPMD 型の並列アプリケーションに対してはよく検討され，発展しているが，複数のアプリケーションを含むワークフローは想定されておらず，その性能情報をデータベースに格納するには難がある．本研究の一部はワークフローアプリケーションに対する性能データベースの設計を含んでいる．

SPMD 型並列アプリケーションではプログラムが 1 つであるのに対し，並列ワークフローの性能解析では複数のアプリケーションが含まれるため，粒度の異なる解析手法が必要になると考えられる．これは特にアプリケーション中の実行パスが一致しないことが原因であると考えられる．本稿では粗粒度なジョブ単位の情報を用い，2 つの性能情報を比較することにより，どのアプリケーションが性能差の原因であるかを解析する手法について述べる．

3. 実装: PARP for Workflow

我々は既に PARP (Performance Analysis with Relative Profiling) という性能比較ツールを提案している．これまでは SPMD 型並列アプリケーションの解析を行ってきた．本節ではこれをワークフローアプリケーションにも対応できるようにするプロトタイプ実装 PARP for Workflow について述べる．

3.1 全体の構成

PARP for Workflow の全体構成を**図 2** に示す．基本的な流れは以下のようになる：

(1) Makefile により記述されたワークフローアプリケーションを GXP Make を用いて実行する．これにより各ジョブの性能情報が出力される．

(2) コマンドラインツールを用い，性能情報を PARP for Workflow のデータベースに格納する．

(3) Web インタフェースを通じて性能を比較し，性能差の原因を解析する．

3.2 PARP for Workflow の動作環境

PARP for Workflow はワークフローアプリケーションの実行とは独立の環境において，Ubuntu Linux 11.04 上で実装を行った．コードは Python をベースとし，データベースには MySQL 5.5.12 Source Distribution を用いている．データベース接続には ODBC を利用している．

3.3 GXP Make の拡張

GXP Make は単独で各ジョブに関する性能情報を出力するが，それ以外にも GXP Make が内部的には持っており，解析で必要な情報が存在する．そこで GXP Make を拡張し，ワーカのリストおよびワークフロー実行開始時刻，ワークフロー全体の実行時間 (Makespan[12]) を別途データベースとして出力している．

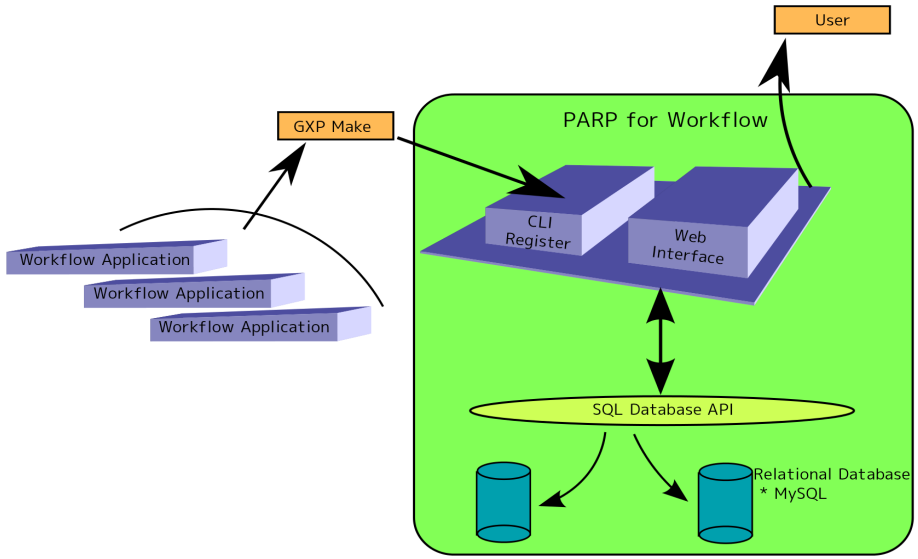

図 2: PARP for Workflow の全体構成
Fig. 2: PARP for Workflow Overview

3.4 データベース設計

ワークフロー向けの比較性能解析においても，SPMD 型並列アプリケーションに向けたものと同じく，性能情報をデータベースに格納することとした．ただし現在までにワークフローの性能を格納するためのデータベース設計を示したものがほとんどないため，ここでは既存の SPMD 型のものを参考にしつつ，独自にデータベースを設計した．

データベース内のテーブル定義と関係性を**図 3** に示す．ワークフローがいくつかのアプリケーションで構成され，そのアプリケーションがジョブという単位で実行されること，また複数の条件下で複数回実行されるということを反映している．また，ワークフロー内のアプリケーションは並列化されていないことを仮定している．この設計は完全ではなく，今後よりよい設計を検討していく．

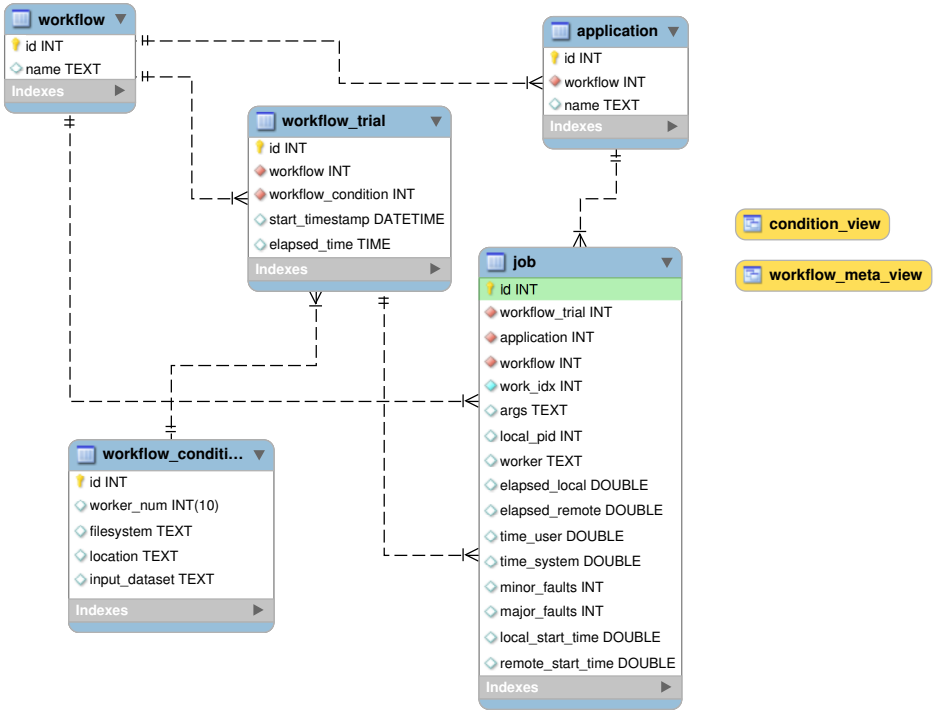

図 3: PARP for Workflow におけるデータベース設計
Fig. 3: Database schemata of PARP for Workflow

3.5 Web インタフェース

Web インタフェースは 2 つの実行結果をグラフ化し，性能差の解析が行えるようになっている．性能解析を行いグラフを出力する上で重要なことは，必要な情報は出力しつつ，不必要な情報は出力しないことである．したがって，インタフェースが出力する情報をユーザが指定できるようになっていることが望ましいと考えられる．

PARP の Web インタフェースではユーザになるべく少ない操作で，必要であれば自由に見たい情報を見られるように設計している．Web インタフェースは本稿執筆時点で実装途中であり，今後改良を重ねながら実装を進める．

表 1: 実験環境クラスタ諸元
Table 1: Specification of Clusters for Experiments

Cluster	Processor	RAM	OS	Filesytem	Interconnection
K	Intel Xeon E5410 2.33GHz	32GB	Debian GNU/Linux Kernel 2.6.26	NFS	1GbE × 2
H	Intel Xeon E5530 2.40GHz	24GB		NFS	10GbE

4. 評　　価

本プロトタイプ実装の評価を行うため，下記に示す実験を行った．

対象としたワークフローアプリケーションは Montage[7] という天文画像のモザイク処理*を行うアプリケーションである．用いた入力データセットは大小 2 種類であり，ここで示すのはワーカが 8 CPUs の場合と 128 CPUs の場合である．実験に用いたサーバ群の諸元を**表 1** に示す．各ノード 8 CPUs であるため，コアの oversubscription を避け，128 CPUs での実験では 16 台のサーバを用いた．8 CPUs の実験では 1 台のみを用い，ノード内並列の際の性能としている．また，小さなデータサイズでの実験はそれぞれ 10 回ずつ実行した平均値を，大きなデータサイズでは 3 回ずつ実行した平均値を，解析に用いた．小さなデータではジョブが 1542 個，大きなデータではジョブが 17585 個生成される．

4.1 例 1: クラスタ K，データサイズ小，8CPUs-128CPUs

まずはクラスタ K において CPU 数が異なるときの解析結果をみる．実行時間は 8 CPUs のとき 223.6[sec]，128 CPUs のとき 179.4[sec] である．PARP for Workflow を用いて生成したグラフを**図 4** に示す．Montage ワークフローは 9 つのアプリケーションにより構成されている．このワークフローではペトリネットで表現したときに分岐がなく，トランジションの重複がない．図 4 では左から順にアプリケーションの実行が行われている．

4.1.1 縦軸の読み方

図 4 には縦軸が 3 種類，5 つの値を示している．最後の数字はそれぞれ 1 が 8CPUs のとき，2 が 128CPUs のときの結果である．`AccumL1`，`AccumL2` が各アプリケーション毎に，各ジョブで要した時間の総和である†．`AppCount1` はそのアプリケーションがワークフロー中で実行された回数である．ここではデータセットが同じなので CPU 数には無関係である．`ElapsedL1`，`ElapsedL2` は各アプリケーションの 1 回あたりの実行時間

* 断片的な画像の接合を行い大きな画像を構成する処理
† 並列実行されていることは考慮せず，単純な和である．

（ElapsedL1 = AccumL1/AppCount1）である．

4.1.2 グラフの解釈

AccumL1，AccumL2 の組み合わせ，ElapsedL1，ElapsedL2 の組み合わせのいずれを見ても 128CPUs のときの方が全体として時間がかかっていることがわかる．これは 128CPUs のときはノード間の並列であり，データの転送を要しているが，8CPUs のときは NFS のキャッシュが効いており，ノード間でのデータ転送が抑制されたことが原因と考えられる．ElapsedL1，ElapsedL2 のみをみると mAdd が長時間を占めているかに思われるが，実際は 1 回しか実行されておらず，全体の実行には寄与しないほど短い実行時間であることがわかる．総計すると AccumL1，AccumL2 から mProjectPP，mDiffFit，mBackground の 3 アプリケーションが大部分の時間を占めていることがわかる．

4.2 例 2: クラスタ K-クラスタ H，データサイズ小，128CPUs

次にワーカの数を固定して 128CPUs を利用することにし，同じデータセットで 2 つの異なるクラスタを利用した場合の性能を比較する．例 1 と同様にして出力したグラフを**図 5** に示す．添字 1 がクラスタ K，添字 2 がクラスタ H を示す．全体の実行時間はクラスタ K で 179.4[sec]，クラスタ H で 80.7[sec] であった．ここで表 1 に示したとおり，プロセッサの周波数には大きな違いがないにもかかわらず，2 倍以上実行時間が異なっており，様々な処理が含まれるワークフローにおいてこの原因は自明でない．

グラフの解釈

図 5 においても例 1（図 4）と同様，mProjectPP，mDiffFit，mBackground の実行時間が支配的であることがわかる．この場合， CPU の性能により 2 倍の性能差が出ることは考えにくく，処理の中でファイルシステムを介したファイル転送が発生していることを踏まえると，クラスタ K では多くのファイル転送のためにノード間通信路が飽和するか，ファイルサーバに負荷がかかりすぎるなど，入出力部分に大きな時間がかかったことにより大きな性能差が発生したのではないかと推測される．

現段階でこの性能差がファイルシステムに起因するかどうかは確かではないが，仮にこの推測が正しかった場合，ファイルのローカリティを考慮したジョブスケジューリングをすることで性能が改善される可能性がある[15),16)]．

4.3 例 3: クラスタ K-クラスタ H，データサイズ大，128CPUs

最後に例 2 において小さな入力に代えて大きな入力を用いた場合の性能比較を示す．全体の実行時間はクラスタ K で 6243.3[sec]，クラスタ H で 3523.0[sec] であった．

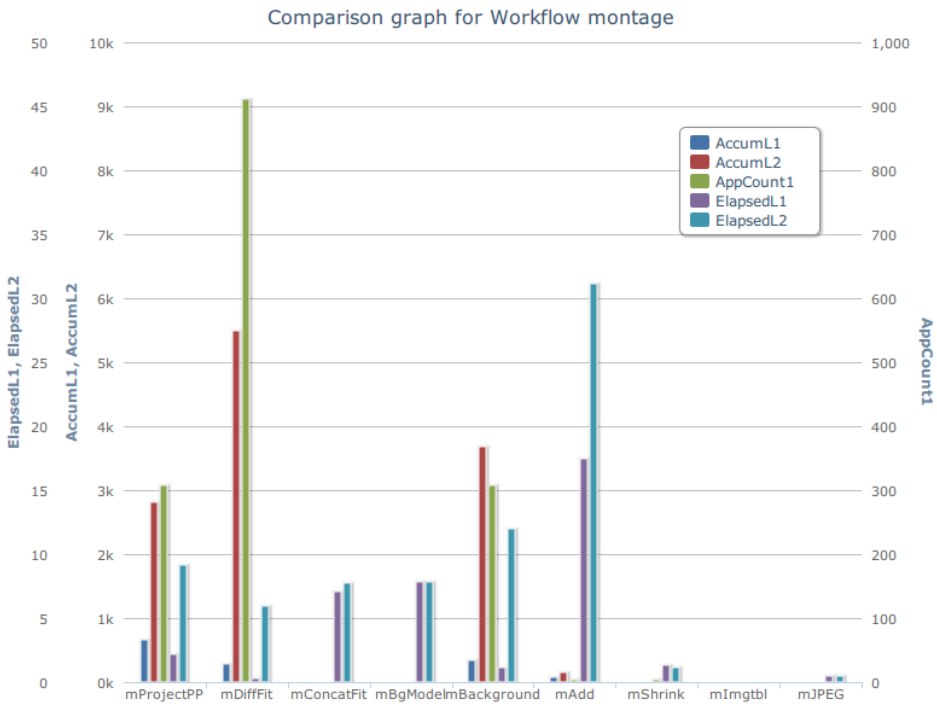

図 4: 例 1: クラスタ K における 8CPUs-128CPUs の性能比較
Fig. 4: Experiment 1: Performance Comparison on Cluster K between 8 and 128 CPUs

グラフの解釈

入力データのサイズが変わっただけなので，基本的な傾向は例 2 のときと同様である．例 1，例 2 では無視できるほど小さかった mAdd の実行時間が無視できない程度に大きくなっているが，これは mAdd の操作はここまでの結果を集計する操作を含んでおり，タスクの分割があまりされず（mAdd は 37 回実行される），入力サイズが非常に大きくなったためであると考えられる．

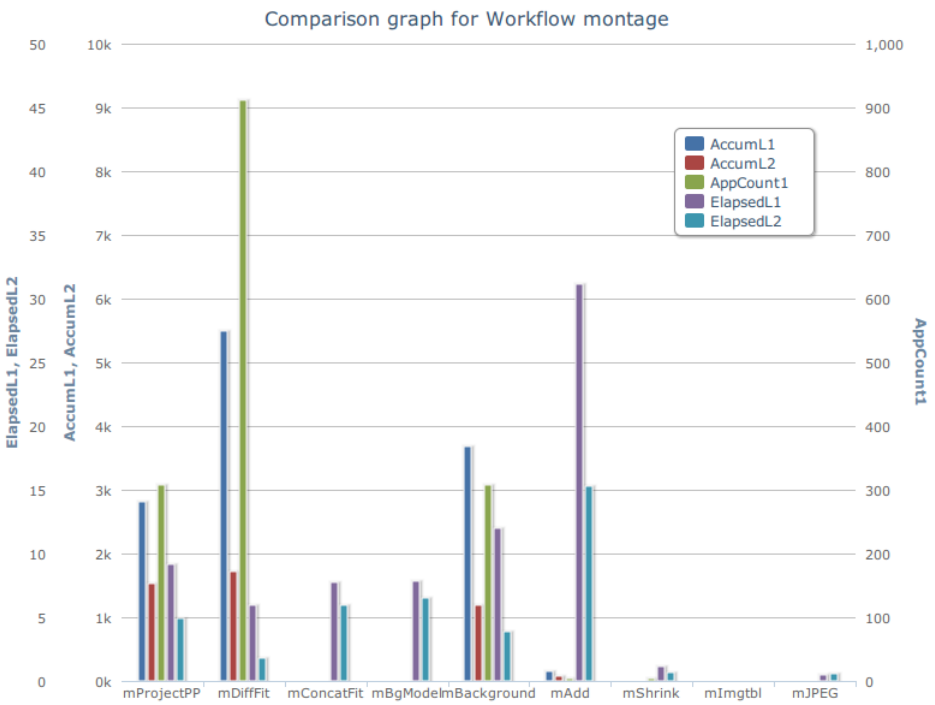

図 5: 例 2: クラスタ K, H における 128CPUs の性能比較（データサイズ小）
Fig. 5: Experiment 2: Performance Comparison between Cluster K and Cluster H for a small dataset

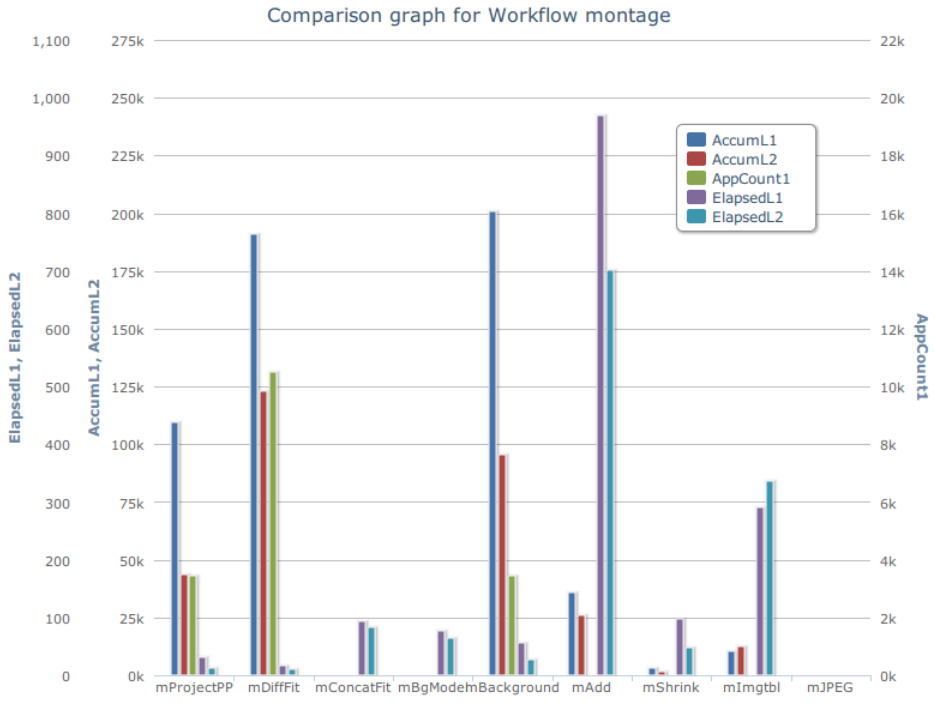

図 6: 例 3: クラスタ K, H における 128CPUs の性能比較（データサイズ大）
Fig. 6: Experiment 2: Performance Comparison between Cluster K and Cluster H for a large dataset

5. おわりに

5.1 まとめ

本稿では異なる環境での並列ワークフローの性能を比較し，最適化に役立てるためのツールとして，従来 SPMD 型並列アプリケーション向けであった PARP のワークフロー向け実装プロトタイプである PARP for Workflow を提案した．実際のワークフローアプリケーションを対象に評価を行い，ワークフローアプリケーションに対しても性能を比較することで問題点の切り分けが可能であることを示した．

5.2 今後の課題

PARP for Workflow はプロトタイプ実装であり，今後検討，実現していくべき点が多々ある．

- **出力する情報の自由な設定** 3 節において示したグラフから分かるように，決まりきった情報だけでは有用な解析結果を得ることはできず，事前にどの情報を出力すればよいかを知っておくことも現時点では困難である．そこで，ユーザの側から自由に出力する情報を設定，選択できるようにしておくことで，解析可能性を担保する方法が考えられる．
- **アプリケーション内部の解析** 現在用いている情報は各ジョブを粒度としている．しかし各アプリケーションの中で何が行われているかを解析することにより，さらに詳細な比較解析が行えると考えられる．単独のアプリケーションのプロファイラは多く存在す

るので，今後これらの情報をユーザに不便を強いることなく利用できるようにすることが重要である．

- **ファイル入出力情報の利用** 現時点でファイル入出力に関する情報は取得できていないが，今後 ParaTrac[1),2)] などのプロファイラを用い，情報を統合することによりさらに性能問題の切り分けが行えるようにしていく．

参 考 文 献

1) Dun, N., Taura, K. and Yonezawa, A.: Fine-Grained Profiling for Data-Intensive Workflows, *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, IEEE, pp.571–572 (2010).
2) Dun, N., Taura, K. and Yonezawa, A.: ParaTrac: A Fine-Grained Profiler for Data-Intensive Workflows, *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ACM, pp.37–48 (2010).
3) Hoffman, J.J., Byrd, A., Mohror, K.M. and Karavanic, K.L.: PPerfGrid: A Grid Services-based Tool for the Exchange of Heterogeneous Parallel Performance Data, *HIPS-HPGC 2005 Joint Workshop on High-Performance Grid Computing and High-Level Parallel Programming Models, in conjunction with IPDPS 2005* (2005).
4) Huck, K.A. and Malony, A.D.: PerfExplorer: A performance data mining framework for large-scale parallel computing, *ACM/IEEE SC 2005 Conference (SC'05)*, pp.41–41 (2005).
5) Huck, K.A., Malony, A.D., Bell, R. and Morris, A.: Design and implementation of a parallel performance data management framework, *International Conference on Parallel Processing (ICPP)*, Oslo, Norway, IEEE Computer Society, pp.473–482 (2005).
6) Huck, K.A., Malony, A.D., Shende, S. and Morris, A.: Scalable, Automated Performance Analysis with TAU and PerfExplorer, *Parallel Computing (ParCo), Aachen, Germany*, Citeseer, pp.1–8 (2007).
7) Jacob, J.C., Katz, D.S., Berriman, G.B., Good, J., Laity, A.C., Deelman, E., Kesselman, C., Singh, G., Su, M.-H., Prince, T.A. and Williams, R.: Montage : a grid portal and software toolkit for science-grade astronomical image mosaicking, *Int. J. Computational Science and Engineering*, Vol.4, No.2, pp.73–87 (2009).
8) Karavanic, K.L., May, J., Mohror, K., Miller, B., Huck, K., Knapp, R. and Pugh, B.: Integrating Database Technology with Comparison-based Parallel Performance Diagnosis: The PerfTrack Performance Experiment Management Tool, *ACM/IEEE SC 2005 Conference (SC'05)* (2005).
9) Li, J., Fan, Y. and Zhou, M.: Performance modeling and analysis of workflow, *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, Vol.34, No.2, pp.229–242 (2004).
10) Merdan, M., Moser, T., Wahyudin, D. and Biffl, S.: Performance evaluation of workflow scheduling strategies considering transportation times and conveyor failures, *2008 IEEE International Conference on Industrial Engineering and Engineering Management*, Ieee, pp. 389–394 (2008).
11) Meseguer, J. and Montanari, U.: Petri nets are monoids, *Information and Computation*, Vol.88, No.2, pp.105–155 (1990).
12) Ostermann, S., Prodan, R. and Fahringer, T.: Trace-based characteristics of grid workflows, *From Grids to Service and Pervasive Computing*, Vol.10, pp.191–204 (2008).
13) Salimifard, K. and Wright, M.: Petri net-based modelling of workflow systems : An overview, *European journal of operational research*, Vol.134, No.3, pp.664–676 (2001).
14) Schulz, M. and de Supinski, B.R.: Practical Differential Profiling, *Euro-Par 2007 Parallel Processing*, Vol.48, pp.97–106 (2007).
15) Shibata, T., Choi, S. and Taura, K.: File-Access Characteristics of Data-Intensive Workflow Applications, *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp.522–525 (2010).
16) Shibata, T., Choi, S. and Taura, K.: File-access patterns of data-intensive workflow applications and their implications to distributed filesystems, *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ACM, pp.746–755 (2010).
17) Song, F., Wolf, F., Bhatia, N., Dongarra, J. and Moore, S.: An algebra for cross-experiment performance analysis, *International Conference on Parallel Processing, 2004. ICPP 2004.*, Vol.1, pp.63–72 (2004).
18) Stratan, C., Iosup, A. and Epema, D.H.: A performance study of grid workflow engines, *2008 9th IEEE/ACM International Conference on Grid Computing*, pp.25–32 (2008).
19) Taura, K., Matsuzaki, T., Miwa, M., Kamoshida, Y., Yokoyama, D., Dun, N., Shibata, T., Jun, C.S. and Tsujii, J.: Design and Implementation of GXP Make – A Workflow System Based on Make, *2010 IEEE Sixth International Conference on e-Science*, pp.214–221 (2010).
20) Truong, H.-L., Dustdar, S. and Fahringer, T.: Performance metrics and ontologies for Grid workflows, *Future Generation Computer Systems*, Vol.23, No.6, pp.760–772 (2007).
21) Valerio, M.D., Sahoo, S.S., Barga, R.S. and Jackson, J.J.: Capturing Workflow Event Data for Monitoring, Performance Analysis, and Management of Scientific Workflows, *2008 IEEE Fourth International Conference on eScience*, pp.626–633 (2008).