Vol.2010-OS-115 No.18
2010/8/4

データ集約的ワークフローの高精度なシミュレーター

崔　　升準[†1]　柴田剛志[†1]
田浦　健次朗[†1]

近年、自然言語処理、天文学、高エネルギー物理学、バイオインフォマティクスなどの科学技術分野における研究では既に大容量計算の必要性は高くなっている。人工衛星のセンサーにより収集されたデータの分析や、生命科学の研究、ポータルサービス提供にも自然言語処理で大容量なデータを扱う計算の需要は高まっている。これに伴ない、大容量計算における並列分散処理でのスケジューリングでは処理時間をいかに短くするかを大きな目的とする。このため大量のデータを共有することも考慮しながら分割したタスクを効率よく割り当てることが求まれる。計算能力の優れたリソースを用いても、タスクが必要とするデータがローカルになければ、そのデータの転送のコストが出てくるためである。

本研究で取り上げているデータ集約的なワークフローは大容量のデータを扱うため、ファイルシステムへのアクセスが数多く発生する。アプリケーションの種類により割合は異なるが、全体の計算時間に高い比率を占めている。一般的にこのようなアプリケーションは大容量なデータから様々なステップで構成されるワークフローにわたって加工されていく仕組みになっており、それぞれのステップ、すなわちタスクの間では中間データからなる依存性を待って結ばれているのが特徴である。本研究では、このようなワークフローアプリケーションのプロファイリングを行った上、その再現及びスケジューリング手法によっての効果を検証できるシミュレーションを実現することを目的とする。

High-Precision Simulation of Data-Intensive Workflow Applications

SungJun Choi ,[†1] Takeshi Shibata [†1]
and Kenjiro Taura[†1]

Parallel distributed processing a large amount of data on grid environment is a challenging issue on e-Science. Each data-intensive workflow application has different rate of file access, but it is not negligible because of file transfer cost. Each step, which we call a job for the term, produces temporal data from one or more input files, and the jobs may have dependency among.

This paper discusses a high-precision simulation on data-intensive workflow application, which could be reproduced the real world execution and adopted for a new policy of scheduling.

1. はじめに

近年、科学技術研究における大容量データ処理の必要性が高まっており、並列分散処理が一般化されている。これに伴い、効率的なスケジューリング問題は数多く議論され、研究されている分野である。関連研究などの紹介先だち、本研究で扱うデータ集約的ワークフローについて説明を述べる。

1.1 ワークフローとは

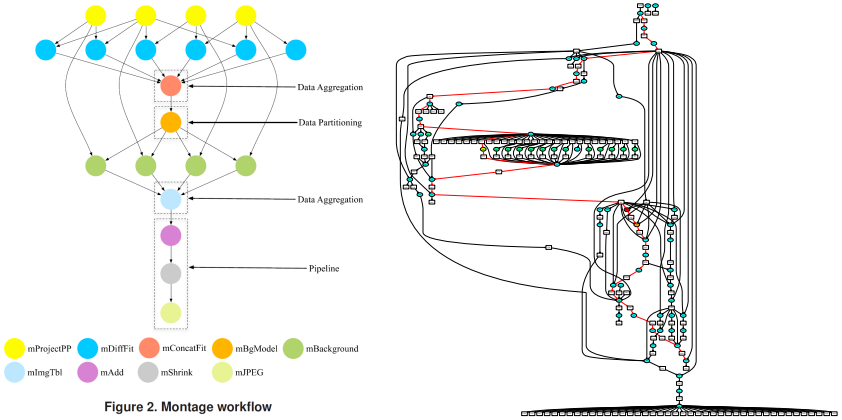
ワークフローとは複数のタスクからなる組み合わせを処理することを意味する。ここでワークフローの個々のタスクはデータからなる依存性を持ち、これによってタスクの実行順序には制約が伴う。図 1 の (a) は複数枚の星の写真をモザイクするシステムのワークフロー[1)] を図示したもので、図 1 の (b) は医学文書から文法的要素を見つけ出し、より豊富かつ、効率的な検索システム[2)] を構築するための医学文書処理システムのワークフローを図示したものである。

図 1 で分かるように、全体のワークフローにわたってデータの依存性がないタスクは並列に処理が可能であって、そうではなく特定のタスクか作成したデータを入力として受け取るタスクはそのデータによる依存性があり、並列化には限界がある。このような制約のもとにいかに賢いスケジューリングをすることで全体の処理時間を短くすることが本研究の目的となる。

2. 関連研究

ワークフローのスケジューリングとは、どのタスクをどの計算機にどのような順序で割り振るかを決めることである。そしてワークフロー全体の実行時間を目

†1 東京大学大学院 情報理工学系研究科 電子情報学専攻
Dept. of Information and Communication Engineering, Graduate School of Information Science and Technology, University of Tokyo



(a) Montage Workflow (b) M2M Workflow

図 1 アプリケーション別のワークフロー

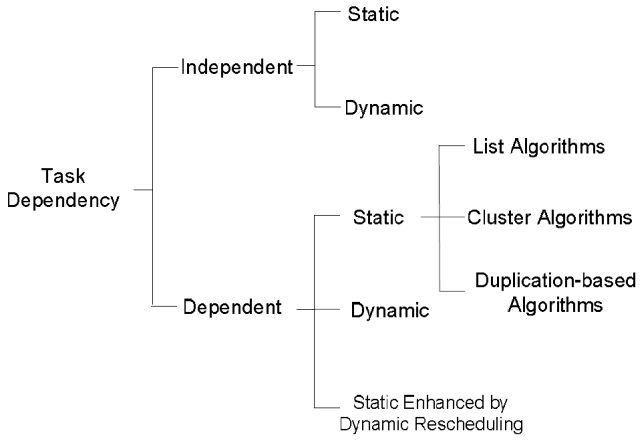


図 2 スケジューリング手法の分類[3)]

的関数とした最適化になる。ワークフローのタスク間の依存性、個々のタスクのコスト、ファイル転送のコストなどの予想値を用いて、効率的なスケジューリングをする手法が多数考えられている。現在研究されているスケジューリングの手法を大きく次の図 2 のように分けることができる。それぞれ、左から、タスク間の依存関係があるかどうか、ワークフローの実行前に全てタスクのスケジューリングを終わらせておくかどうかである。[?)]

2.1 GXP Make

GXP Make は並列グリッドシェル GXP の並列分散処理系で、Python 言語で実装されており、軽量で導入コストが低いツールである。[4)]

GXP Make は通常の GNU Make 仕様で記述された Makefile を GNU Make に解析させ、発生したジョブを GXP 側で受取り、予め確保しておいたホスト上にジョブを割り当て、リモート実行する仕組みをとっている。そして Make の並列実行オプション (-j) を付けることで、並列分散処理が可能になっている。

GXP Make にはモニタリングシステムがあり、HTML 上で、並列度・マスタ側の loadavg 値、等の情報や、ジョブの実行状況をリアルタイムに観覧することができる。Make の性質上基本途中までの実行は Make 側で依存性関係の解析により失敗時点からの再実行が可能であり、GXP Make ではそれぞれのジョブの失敗状況に応じて失敗したジョブを他の計算資源に再度割り当て実行することでフォルトトレランスを実現している。さらに、計算資源の参加脱退もサポートしており、GXP Make で既に何らかの不具合によりリモート実行が不可能になった計算資源を切り捨てることができ、新しく使用可能になったホストを計算資源として加え、処理の並列度を上げることが可能になっている。

GXP Make はすべての計算資源のファイルシステムが共通だという仮定において実行されるため、ノード間ではクラスタ内 NFS 共有システムのように、確保した全ノードに対し分散共有ファイルシステムが用意されてなければいけない。

本研究にすすめるにあたってワークフロー実行の並列分散処理系として GXP Make を導入し、実験を進めた。

2.2 ジョブの適応的な実行時間の予測

Glasner らはジョブの実行時間を適応的に予測する手法を提案しているが、ジョブの実行予想が現実問題に適用できるほどの精度で求められる理由は主に二つの観点から述べている。[5)]

ユーザ 計算リソースを実際、アプリケーションを実行するユーザとしては、効

率的なスケジューリングだけでなく、例え、AmazonEC2 のような課金制のリソースを利用する場合においての使用コスト削減面でも実行時間などの予想は重要な役割をする。

管理者　また、リソースを使われる側としては、システムの管理及び運用上のコスト対比、利用率、つまり、ジョブがリソース全般においての空きがないよう実行されていて欲しい。

Glasner らは従来の予想手法でのジョブの実行履歴の統計的な解析からの特性だけでなく、ユーザ及びユーザグループの利用パターンを考慮する予想手法を提案している。つまり、ユーザ、ホスト、ジョブのクラスタリングに多数の手法を適用し、類似性の検知によってジョブ実行の予想を図る。

ユーザの利用パターンなどのクラスタリングには以下を考慮している。[6)]

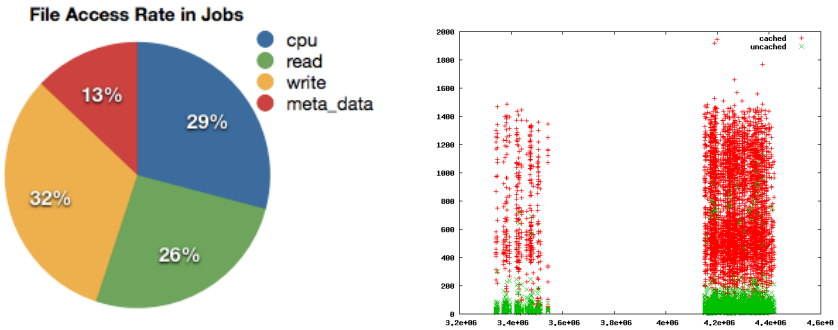
- ユーザは似たようなジョブを異なる環境で実行している。つまり、類似するジョブでも実行時間は異なる。
- 類似する特性のジョブが異なるワークフローに含まれているかもしれない。
- 異なるワークフローでお互い混ざっているかもしれない。
- ユーザは決まった時間に必ずジョブを投入するとは限らないため、再現の検知は難しい。
- 各ジョブは複数の属性をもつ。
- 異なる特性のジョブ間は依存性があるかもしれない。

これらを考慮した上でのクラスタリングされた実行履歴データから予想対象となるジョブと属性を共有するジョブ履歴データを抽出する。そして各クラスタにおける予想器に入力として与え、予想時間を計算して最終的にジョブの予想実行時間を求めている。

3. 実験及び評価

3.1 実験環境及び方法

本研究のために実験環境として InTrigger の tohoku 拠点を用いた。[7)] 分散ファイルシステムとして NFS を用い、3.2 節で述べるように、ワークフローのプロファイリングを行った。ワークフロープロファイリングに用いたツール Logfuse は FUSE API を利用し、GXP Make のジョブのプロファイリングを行う。[8)]Logfuse によりワークフローのジョブ間の依存性やファイルアクセス情報を取得し NFS においてのファイルアクセスの振る舞いの解析を 3.3 節で行った。



(a) Montage data2 における全体のジョブでのファイルアクセスの割合　　(b) NFS 環境におけるファイルアクセスのキャッシュ効果

図 3　NFS 上での Montage をプロファイリングした結果の解析

3.2 ワークフロープロファイリング

本研究で扱うワークフローがデータ集約的な特性からワークフローの実行におけるファイルアクセスのコストはずいぶん大きい。図 3 の (a) は全ジョブの実行時間の中で、それぞれ CPU 時間、ファイルの読み込みの時間、書き込みの時間、Meta-data 処理の時間の割合を図示している。ここでファイルの読み込み時間の割合が 26%を占めていてスケジューリングによっては全体の実行時間を大幅削減することができると考えられる。その理由を 3.3 節での解析結果から述べる。

3.3 ファイルアクセス

複数ノードでの実行では共有分散ファイルシステム上で実行するためファイルシステムの種類によるファイルアクセスの特性が変わってくる。図 3 の (b) は NFS 上で InTrigger プラットフォームの tohoku 拠点内 10 ノードを確保し、GXP Make による Montage ワークフローの実行の際、Logfuse を介しワークフローのプロファイリング行い、そのログデータから NFS のキャッシュ効果を解析した結果を図示した。

それぞれのジョブに対して、ジョブの入力ファイルの中で他のジョブで既に同ノードで読み込みされているファイルをキャッシュされているとし、横軸をファイルのサイズ、縦軸を読み込みのバンド幅に赤い点に図示している。また、ジョブ

が実行されるノードで読み込まれてないファイルをキャッシュされてないとし、緑色で図示している。

NFS では一度読み込みがあったファイルに対してはキャッシュ効果があって、同環境を前提に新しいスケジューリング手法によるシミュレーションにおいてファイルの読込みの状況を反映する必要があると言える。

4. おわりに

本研究では一つの拠点内で NFS 上でのワークフローのプロファイリングデータを元にファイルアクセスの度合いや振る舞いを解析している。これを基にシミュレーションによる再現を行ない、他のスケジューリング手法の評価を行う。さらに他のファイルシステムにおける評価も進めていく予定である。

参考文献

1) Montage Workflow. http://montage.ipac.caltech.edu/
2) Medie search engine. http://www-tsujii.is.s.u-tokyo.ac.jp/medie/index.cgi
3) Fangpeng Dong and Selim G. Akl, Technical Report No. 2006-504 Scheduling Algorithms for Grid Computing: State of the Art and Open Problems, 2006
4) K. Taura. Gxp: An interactive shell for the grid environment, Procs Innovative Architecture for Future Generation High-Performance Processors and systems, pp.59-67, 2004.
5) C. Glasner and J. Volkert. Adaptive run-time prediction in heterogeneous environments. Proceedings of the 18th ACM international symposium on High performance distributed computing (HPDC'09), Garching, Germany, 2009.
6) C. Glasner and J. Volkert. An architecture for an adaptive run-time prediction system. In Proceedings of the 7th International Symposium on Parallel and Distributed Computing (ISPDC' 08), Krakow, Poland, 2008.
7) InTrigger, distributed grid cluster platform. http://www.intrigger.jp/wiki/index.php/InTrigger
8) Profiler for GXP Make. http://www.intrigger.jp/wiki/index.php/Profiler_for_GXP_make