1H-04

データのアクセス頻度に着目した Hadoop 分散ファイルシステムの性能向上

山本 景† 川原 純† 笠原 正治†
†奈良先端科学技術大学院大学

1 はじめに

Apache Hadoop は 分散処理アルゴリズム MapReduce [2] のオープンソース実装である．MapReduce や Hadoop は，計算や解析などのジョブをタスクと呼ばれる小さな単位に分割し，システムを構成するノードに分散させ処理を行う．Hadoop 上で処理されるデータは，通常，Hadoop 分散ファイルシステム（HDFS）上で各ノードに分散配置される．HDFS は，データをブロックという単位に分割し，いくつかの複製を生成する．HDFS に関する研究は [3] などがある．

Lee ら [4] は，タスクを処理するノードとタスクが必要とするブロックの位置関係により，処理速度に大幅な変化が生じ得ることを明らかにしている．タスクを処理するノードがデータブロックを持たない場合，ネットワーク通信によってそのブロックを取得する必要があるため，両者はネットワーク上で近い位置にあることが望ましい．HDFS のデフォルトの設定では，データは固定長のブロックに分割され，ファイルの使用頻度にかかわらず，ブロック 1 個につき 2 個の複製が生成される．ブロックの複製数が多い方が，タスクが必要とするブロックとタスクが近くなる可能性が高まる．本研究では，ファイルのアクセス頻度に基づいて，ブロックの複製個数を変える方式を提案し，システム全体のスループットの向上を目指す．

2 準備

本研究では，分散環境における，ブロックの配置方式のみを考察する．そこで，Hadoop 環境を抽象化した，以下で説明する単純化したモデルを考える．データセンタには複数ラックが存在し，各ラックには定められた個数のノードが存在する．各ノードはデータ保存のための定められた大きさのストレージを持つ．Lee らは，タスクを処理するノードとタスクが必要とするブロックの位置関係を (i) ノードとブロックの位置が同じ（ノード局所性），(ii) ノードとブロックが同一ラック内（ラック内局所性），(iii) ノードとブロックが別ラック（ラック外局所性），の 3 種類に分類している [4]．本研究は，あるタスクがブロックを取得する際に，それぞれ定数 T_1, T_2, T_3 $(T_1 < T_2 < T_3)$ の時間がかかると仮定する．

本研究では，データセンタ内に構成された HDFS 上に配置されたファイルについて，ファイル内の単語の検索を行うジョブを想定する．1 つのジョブでは 1 つのファイル内の 1 つの単語を検索する．ファイルはいくつかのブロックに分割されるため，ジョブはブロックの個数のタスクに分割され，各タスクが 1 つのブロック内の単語検索を担当する．単語検索には固定の検索時間 T_{s} がかかると仮定する．

データセンタはスケジューラを 1 つ持ち，スケジューラには，ファイル検索要求のジョブが次々に到着する．スケジューラはジョブをタスクに分割し，各タスクにファイルを構成するブロックの 1 つを割り付け，タスクキューの末尾にそれらを追加する．各ノードは同時に 1 つのタスクだけを処理できると仮定し，あるタスクの処理が終われば，タスクキューの先頭のタスクを取り出し，そのタスクの処理を始める．タスクの処理はデータ取得と検索の 2 つからなる．タスクの実行時間はデータ取得時間と検索時間の和 $T_i + T_{\mathrm{s}}$ であり，$i(= 1, 2, 3)$ の値はタスクとタスクの担当するデータの位置関係で決まる．ネットワークの混雑は考慮しない．ジョブ内のタスクがすべて終了したとき，そのジョブが終了したとみなす．

3 提案方式

デフォルトのブロック配置方式では，各ブロックについて，その複製を同じ個数作成する．1 個目はランダムに配置ノードを選び，2 個目は 1 個目と同じラックで異なるノードをランダムに選び，3 個目以降は他

An improvement of Hadoop distributed file system based on the access frequency of data
Kei Yamamoto†, Jun Kawahara†and Shoji Kasahara†
†Nara Institute of Science and Technology
{yamamoto.kei.yh2, jkawahara, kasahara}@is.naist.jp

と異なるラックのノードをランダムに選ぶ．以上の手順を，ノードのストレージに空きがある限り行う．すべてのブロックについて，複製の数は同じになるようにする．

提案方式では，複製の数を，ファイルのアクセス頻度に比例する割合になるように調整する．ブロックはランダムに配置する．

4 シミュレーション

以上で述べた設定でモンテカルロシミュレーションによる数値実験で両方式の性能を比較する．1 ステップごとにジョブが複数個到着する．5000 ステップ目までジョブが到着するとし，ジョブが到着してから処理が完了するまでのステップ数（平均待ち時間）を評価する．ブロック取得時間は $T_1 = 1, T_2 = 10, T_3 = 100$（ステップ）とし，検索時間は $T_\mathrm{s} = 5$ とする．ラック数は 20, ノードの総数は 1000 である．1 ノード当たりのブロックの最大保持数は 20 とする．実装言語は Python 2 である．

Ananthanarayanan ら [1] は，データセンタにおけるファイルサイズとファイルの個数に関する統計データを示している．本研究では，それに従い，ファイルサイズ x が発生する確率密度関数は $f(x) = (9 \times 10^6)x^{-1.6}$ であるとし，シミュレーションの開始時に，f に従ってサイズが決められたファイルが存在すると仮定する．それらのファイルのブロックを各方式で配置する．ファイルのアクセス頻度は統計 [1] を参考に，ファイルをサイズに応じて 3 つのグループに分割し，それぞれのグループに属するファイルのアクセス頻度の比は小さなファイルから順に 2:1:3 であると仮定する．

デフォルト方式と提案方式について，ジョブ到着の頻度を変えたときの，平均待ち時間の変化を描いた図を図 1 に示す．紙面の制約のため，その他の結果は発表時に紹介する．

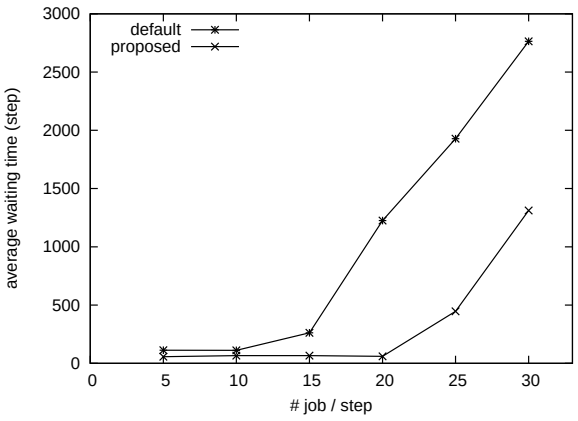

図 1: ジョブ到着の頻度と平均待ち時間の変化

5 まとめと今後の課題

ブロックの配置方式がジョブの平均待ち時間に影響を及ぼすことが確認できた．実験結果はパラメータ設定によって大きく変わり得るため，パラメータの設定を注意深く行う必要がある．デフォルト方式は，ジョブの処理時間だけではなく，ファイルの可用性やネットワーク故障も考慮して決められた方式であると考えられるが，本稿では，ジョブの処理時間のみを考慮した．

今後は，実際の Hadoop へ実装を行い，処理時間が改善することを示す必要がある．今回想定した方式では，タスクのノードへの割当スケジューリングについては考えなかったが，この点も考慮することで，性能がより改善できる可能性がある．

参考文献

[1] G. Ananthanarayanan, A. Ghodsi, A. Warfield, D. Borthakur, S. Kandula, S. Shenker and I. Stoica, “PACMan: Coordinated Memory Caching for Parallel Jobs,” Proc. of the 9th USENIX Symposium on Networked Systems Design and Implementation, pp. 267–280, 2012.

[2] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” Proc. of the 6th Conference on Symposium on Opearting Systems Design & Implementation, vol. 6, p. 10, 2014.

[3] A. Higai, A. Takefusa, H. Nakada and M. Oguchi, “A Study of Effective Replica Reconstruction Schemes for the Hadoop Distributed File System,” IEICE Transactions on Information and Systems, vol. E98.D, no. 4, 872–882, 2015.

[4] J. Lee, J. Chung and D. Lee, “Efficient Data Replication Scheme based on Hadoop Distributed File System,” International Journal of Software Engineering and Its Applications, vol. 9, no. 12, 177–186, 2015.