Hadoop のノード削除時のレプリカ生成の高速化

日開 朝美　(指導教員：小口 正人)

1　はじめに

近年情報が爆発的に増加し，汎用のハードウェアを用いて大規模データの高度な集約処理を行う分散ファイルシステムに注目が集まっている．本研究では，Apache Hadoop の基盤技術である Hadoop Distributed File System(HDFS) に着目した [1][2]．Hadoop はスケールアウトするシステムであり，システム規模に応じた運用や故障の発生などにより，クラスタからノードを切り離すことが想定される．HDFS では一般に複数のレプリカを保持し，クラスタから一部のノードが切り離されると他のノードにレプリカを自動的に生成する．しかし，この過程が長くなると HDFS の性能低下につながり，その高速化が重要である．本研究では，ノード削除時のレプリカ生成を高速化する手法を提案し，これを実装した評価実験を行い提案手法の有効性を検証する．

2　ノード削除時のレプリカ生成の流れ

2.1　Hadoop Distributed File System

HDFS は，ファイルのメタデータやクラスタ内のノード管理を行う一台の NameNode と，実際にデータを格納している複数台の DataNode からなる．ファイルを最小単位であるブロックに分割して DataNode 間で分散して保存することで，耐故障性と対多クライアントでの高いパフォーマンスを維持している．

2.2　レプリカ生成の流れ

HDFS はクラスタからノードを切り離す際，該当ノードが保持しているデータのレプリカを残りのノードに移行することで，指定したレプリカ数を維持する．ノード削除処理が実行される場合，レプリカ生成処理が行われる．レプリカ生成はブロック単位に処理が進み，まず NameNode が定期的にレプリカ生成の指示をそれぞれの DataNode に転送する．DataNode は受けとった指示をもとに，レプリカ生成先の DataNode へデータの転送を始める．生成先でデータの複製が完了すると，生成元の DataNode に生成完了の ACK を転送する．この時 NameNode が定期的に DataNode に指示する転送間隔は `replicationRecheckInterval` で定義され，転送レプリカブロック数は，クラスタに接続されている DataNode 数*`REPLICATION_WORK_MULTIPLIER_PER_ITERATION` で定義される．DataNode が生成完了の ACK を受け取らない状態のまま，レプリカ生成先の DataNode へ転送できるブロック数は `dfs.max-repl-stream` で定義される．`replicationRecheckInterval` と `REPLICATION_WORK_MULTIPLIER_PER_ITERATION` は，FSNamesystem.java の ReplicationMonitor クラスで定義されている変数であり，`dfs.max-repl-stream` はプロパティで変更可能な変数である．各変数のデフォルト値は表 1 のようになっている．

表 1: 各変数のデフォルト値

変数	デフォルト値
`replicationRecheckInterval`	3
`REPLICATION_WORK_MULTIPLIER_PER_ITERATION`	2
`dfs.max-repl-stream`	2

3　基本性能評価

3.1　実験環境

本実験ではレプリカ生成に関する評価を行うために，ブロックサイズおよび同時に処理するデータ量の変化がレプリカ生成処理のスループットに与える影響を調査する．ブロックのレプリカ数を 3 とし，1 台の DataNode をクラスタから切り離す際のスループットを示す．ここでスループットは以下のように定義した．

$$\text{スループット (MB/sec)} = \frac{\text{脱退及び削除ノードが保持しているデータ量 (MB)}}{\text{データの複製が開始してから完了するまでの時間 (sec)}}$$

ローカルクラスタ上で Hadoop-1.0.3 をインストールしたマシン 7 台を用いた．そのうち 1 台を NameNode とし，残りの 6 台を DataNode とした．マシンのスペックは全て同一で表 2 に示す通りである．

表 2: マシンスペック

OS	Linux 2.6.32-5-amd64 Debian GNU/Linux 6.0.4
CPU	Quad-Core Intel(R) Xeon(R) CPU @ 1.60GHz
Main Memory	2GB
HDD	73GB SAS × 2(RAID0)
RAID Controller	SAS5/iR

3.2　同時転送ブロック数を変化させた実験

ブロックサイズ 16，32，64，128，256MB に対して，レプリカ生成時の NameNode と DataNode の同時転送ブロック数に関連する，`REPLICATION_WORK_MULTIPLIER_PER_ITERATION` と `dfs.max-repl-stream` の 2 つの変数を共に同じ値 α とし，α を 1〜8 に変化させてノード削除を行った際のスループットを図 1 に示す．図 1 より，α が小さいとき，すなわち同時転送ブロック数が少ない場合，ブロックサイズが小さい時にはスループットが低くなる．これはブロックサイズが小さいと DataNode はレプリカ生成の処理が完了して，NameNode から定期的に送られてくるレプリカ生成の指示を待っている状態が生じるからと考えられる．またこの状態にある間は，α に比例してほぼ線形にスループットが増加している．

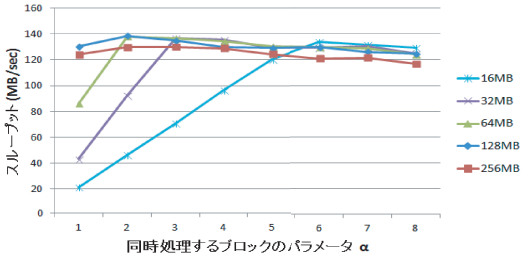

図 1: 同時処理するブロック数に応じたスループット

また本実験で最もスループットが高かったブロックサイズが 128MB，α が 2 の時の 5 台の DataNode のトータルスループットと 5 台各々の DataNode のスループットの変化をそれぞれ図 2 と図 3 に示す．横軸は時間 (sec) で縦軸の正方向が受信のスループット (MB/sec)，負方向が送信のスループット (MB/sec) である．図 2 より，安定したスループットが維持されていないことが分かる．また図 3 より，同時刻のスループットを DataNode

間で比較すると送信，受信共にスループットの値が大きく異なる時刻が多く見られることから，ノード間においてスループットにばらつきがあることが分かる．

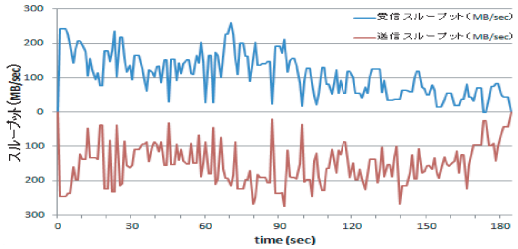

図 2: トータルスループットの変化

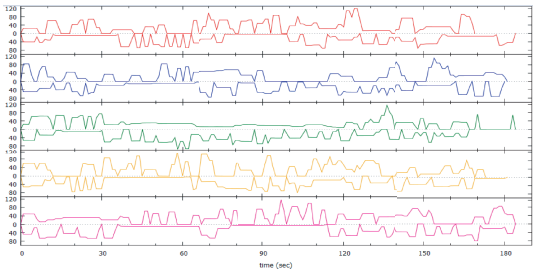

図 3: 各々の DataNode のスループットの変化

4　生成先を考慮した制御手法の提案と評価

前述の実験結果より，レプリカ生成処理のスループットは一定の値が維持されておらず，効率の良い処理が行われていないことが分かった．従って効率の良い処理を実現することでレプリカ生成処理の高速化が見込める．HDFS では，同一ラックに配置されたレプリカの生成先は該当のレプリカを持っていないノードからランダムに選出される．そこで，レプリカ生成処理の高速化に向けて，まずはレプリカ生成先を制御する手法を提案し，制御前後でスループットを比較して有効性を検証する．

4.1　提案手法

データ通信に関してノードをリング状に配置し，そのリング構造に従って一方向に通信を行うと効率が良いことが一般に知られている．そこで本提案手法では，ノードがリング状に配置されていると仮定し，(1) レプリカ生成先を次のノードに指定する．もしそのノードがすでにレプリカを持っている場合は，(2) さらに次のノードをレプリカ生成先にする (図 4)．このようにレプリカ生成先を制御することで，多数のノードと通信する事を回避し効率の良いデータ処理を目指す．

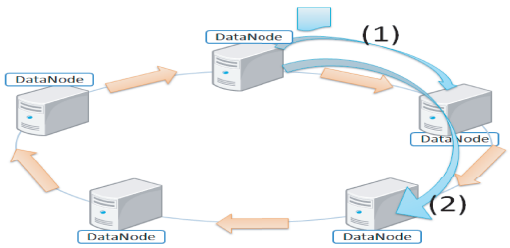

図 4: 提案手法のレプリカ生成先

4.2　提案手法の評価

3.2 節と同様の実験を提案手法に関して行い，ブロックサイズ毎の制御前後のスループットの測定結果を図 5 に示す．図 5 より，DataNode が生成完了の ACK を受け取らない状態のまま，レプリカ生成先の DataNode に転送できるブロック数以上のレプリカ生成の指示を NameNode から受け取っている時には，全ての場合で提案手法のスループットが向上している．16，32，64MB においてαが小さい時には，制御前後のスループットに変化がないが，これは 3.2 節で述べたように DataNode は NameNode からのレプリカ生成の指示を待っている状態であると考えられることから，妥当な結果である．また提案手法は，ブロックサイズが大きいほどスループットが向上していることが確認できる．ブロックサイズが大きい方が 1 つのブロックのレプリカ生成にかかる時間が長く，複数ノード間での通信効率を向上させる提案手法の効果が大きいためである．

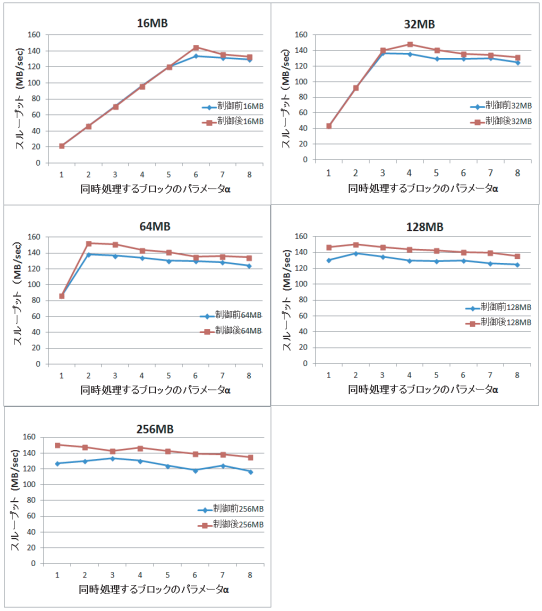

図 5: ブロックサイズ毎の制御前後のスループット

5　まとめと今後の課題

分散ファイルシステムの一つである HDFS 上で，ノード削除時のレプリカ生成に関する基本性能の評価を行った．ブロックサイズが小さくかつ同時転送ブロック数が少ないと，NameNode からのレプリカ生成の指示待ち状態が生じスループットが低くなること，およびレプリカ生成処理は，効率の良い処理が行われていないことを確認した．またレプリカ生成を高速化するためにレプリカ生成先を制御する手法を提案し，提案手法による高速化の有効性を示した．

今後の課題としては，提案手法によりスループットが向上することは確認できたが，一定の安定したスループットの維持に関する議論には至らなかったので，今後検証していきたい．また更なるレプリカ生成の高速化の制御手法の提案に向けて，生成元の制御やラックを意識した制御を行っていきたい．

参考文献

[1] Tom White(著), 玉川竜司，兼田聖士 (訳):Hadoop, オライリー・ジャパン,2010

[2] Dhruba Borthakur, ”HDFS Architecture,” 2008 The Apache Software Foundation.

[3] 日開朝美，竹房あつ子，中田秀基，小口正人:Hadoop のノード脱退時および削除時のレプリカ生成に関する一考察，DEIM2013,2013 年 3 月発表予定