1K-05

NVMe SSD と IEEE802.3an/bz を考慮した分散ファイルシステムの提案

比嘉 隆貴 † 市川 嘉裕 † 山口 智浩 †
奈良工業高等専門学校 情報工学科 †

1 背景と目的

並列分散処理は，複数台のコンピュータを利用し，大量のデータを高速に処理する手法であり，これには複数のコンピュータから同一ファイルを取得する分散ファイルシステムが利用される．Ceph[1] では，HDD や SSD などの記憶装置の性能に応じて重みを変更することが可能で，デフォルトではハードウェア情報などから重み付けがされる．しかし，記憶装置の詳しい性能やネットワーク性能等は考慮されない．HDFS[2] も同様に，詳しい記憶装置やネットワーク性能等が考慮されない．そのため，これらのファイルシステムにおいて，現代のコモディティなコンピュータでは最大限の性能を発揮できていない．そこで，本研究では，事前に記憶装置やネットワーク性能を評価し，重み付けを行う分散ファイルシステムの新たな機構を提案し，スループット性能の向上を図る．具体的には，既存の分散ファイルシステムをベースに参照の局所性に関してキャッシュを適用し，NVMe, IEEE802.3an/bz を考慮する．本稿では，機構の設計と実装，そして性能評価について述べる．

2 提案システム

2.1 概要

提案システムでは，図 1 のように Worker，Shadow Controller，Controller の 3 つの要素で構成される．

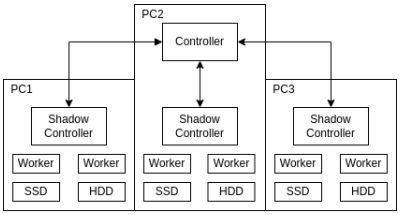

図 1 本研究の分散ファイルシステムの概略図

ファイルは複数のチャンクと単一のチャンクヘッダーに分割され，前述の要素間で送受信される．チャンクヘッダーは図 2 のように，先頭 32 バイトに，32 バイト以降のデータの Sha256 のチェックサムを保持し，次の 8 バイトでファイルのチャンク数を保持する．40 バイト以降は可変長であり，各チャンクのチェックサムを順番に保持する．また，チャンクは図 3 のように，先頭 32 バイトに，32 バイト以降のデータの Sha256 のチェックサムを保持し，次の 8 バイトでデータサイズを保持し，そのデータサイズ分を 40 バイト以降に格納する．なお，データサイズは 64MB より小さい．

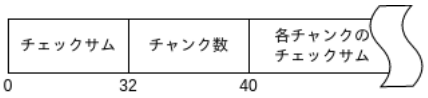

図 2 チャンクヘッダー

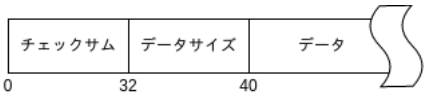

図 3 チャンク

また，分散ファイルシステムに指定されたパスは UUIDv5/OID によって UUID に変換し，チャンクヘッダー及びチャンクのファイルの名前となる．また，チャンクのインデックスを UUID の後ろに付与する．

2.2 Worker

Worker はチャンクの読み書きを担っており，データを書き込む際チェックサムを生成する．また，10GB のファイルの読み書き速度を定期的に計測し，式 (1) によって，スコアを算出する．

$$score = x \times \min((a \times 1.02 + b) \div 2.01, c) \quad (1)$$

x を空き容量，a を読み込み速度，b を書き込み速度，c を Shadow Controller から通知される速度とする．なお，速度の単位は KB/s である．また，チャンクの読み書きは ext4 などの既存のファイルシステム上で行う．

2.3 Shadow Controller

Shadow Controller はチャンクヘッダーを保持し，キャッシュの管理を行う．キャッシュは，ファイル単位で最大 128MB 保持し，全体で 16GB までメモリに保持する．その後，スコアの高い Worker へチャンクを書き込む．なお，スコアが均等になるように，定期的に読

Distributed File System Based on NVMe SSDs and the IEEE802.3an/bz Standard

† Ryuki Higa, Yoshihiro Ichikawa, Tomohiro Yamaguchi・Department of Information Enginnering, National Institute of Technology, Nara Collage

表 1　NVMe と HDD 混合，10Gbps ネットワーク環境下におけるファイル単体の読み込み速度 (MB/s)

ファイルサイズ	1GB				10GB				300GB			
システム	最大	平均	最小	偏差	最大	平均	最小	偏差	最大	平均	最小	偏差
Ceph	733.79	690.93	644.56	27.45	790.46	719.97	664.76	37.43	525.81	485.80	453.97	21.14
HDFS	480.08	448.67	422.47	17.47	505.15	473.62	446.05	15.64	490.06	446.34	409.00	24.13
提案システム	926.43	791.57	681.29	77.90	796.70	721.00	650.73	47.74	543.05	492.35	448.08	28.38

表 2　NVMe と HDD 混合，10Gbps ネットワーク環境下におけるファイル 100 個の読み込み速度 (MB/s)

ファイルサイズ	10MB				100MB				3GB			
システム	最大	平均	最小	偏差	最大	平均	最小	偏差	最大	平均	最小	偏差
Ceph	714.88	676.61	634.19	22.68	767.28	705.77	652.67	32.89	526.03	487.51	452.15	21.70
HDFS	378.21	358.24	341.77	9.79	414.51	393.73	375.94	11.55	414.87	383.70	355.67	16.40
提案システム	959.45	830.82	699.49	77.14	816.56	739.93	665.99	45.26	584.96	525.92	477.40	31.19

み書き日時が古いチャンクからスコアの低い Worker へチャンクを移す処理を担う．

2.4　Controller

Controller は Shadow Controller とのネットワーク速度を定期的に測定し，通知する．ファイルの読み書きも担っており，Worker のスコアに応じて，Shadow Controller へ 3 つチャンクを書き込むよう通知する．スコアが一番高い Worker を持つ Shadow Controller に対しては，キャッシュを持つように通知する．また，チャンクヘッダーを生成し，保持する．なお，Contoller が突然停止した場合でも，Shadow Controller が持つ情報で新たに Controller を生成することが可能である．

3　実験

2 章にて提案した手法を Rust 言語を用いて実装した分散ファイルシステム，Ceph(16.2.7) そして HDFS(3.3.1) においてファイルの読み込み性能の比較を行った．Ceph や HDFS はドキュメントに記載されている方法でインストールした．実験環境は Corei9-10900K とメモリ 64GB のコンピュータ 1 台, Corei9-11900K とメモリ 32GB のコンピューター 2 台で構成されたクラスタで, すべてにおいて, 最大読み書き速度約 2400MB/s の NVMe SSD と最大読み書き速度約 200MB/s の HDD, そして IEEE802.3an/bz 対応の NIC を搭載している．システムには各記憶装置 200GB を割り当て, 計 1200GB の中で実験を行った．なお，提案システムと HDFS の評価は ext4 ファイルシステム上で行った．

4　結果と考察

3 章で述べた環境において，ファイルサイズ及びシステムごとに 100 回ずつ測定を行った．それらの最大値，平均値，最小値，そして標準偏差について計算を行い，1GB，10GB，300GB のファイル単体で測定した結果を表 1 に，10MB，100MB，3GB のファイル 100 個で測定した結果を表 2 に示す．

表 1, 表 2 より, 大きなファイル 1 つ及び, 小さいファイル複数の場合でも提案システムに優位性があった．ただし，提案システムでは，偏差が大きく速度が安定していない．原因としてキャッシュがヒットしない，チャンクを移す処理，環境の測定，などの要因が読み込み時に被り，速度が低下していると考えられる．

5　まとめと課題

本稿では，分散ファイルシステムの新たな機構を提案し，開発を行った．そして，スループット性能について検証を行い，3 章で述べた環境において優位性があった．ただ，事前に環境に関して測定すること，Ceph や HDFS に比べて機能が著しく乏しく，オーバーヘッドが少ないことを考慮すると，必ずしも優位性があるとは言い切れないと考える．今後は，耐障害性やディスクを追加したときの性能等において正確で大規模なさらなる検証を行い，本分散ファイルシステムの欠点を洗い出す．また，分散ファイルシステムの安定化そして性能向上を図るため，さらなる改良及び実装を行う．

参考文献

[1] Ceph authors and contributors, Ceph, https://docs.ceph.com/en/pacific/, 2022-01-03 参照

[2] Apache Software Foundation, Apache HDFS 3.3.1, https://hadoop.apache.org/docs/r3.3.1/, 2022-01-03 参照