IPFS におけるデジタルアート流通のための最適化機構

吉川　綾乃†　　松原　克弥†

† 公立はこだて未来大学　〒 041–8655 北海道函館市亀田中野町 116 番地 2
E-mail: †{g2121060,matsu}@fun.ac.jp

あらまし　デジタルアートの安全なオンライン流通を可能にした NFT マーケットの多くは，作品データをオフチェーン上で保管するために，IPFS を採用している．主要なデジタルアートのひとつである，写真やイラストなどのデジタル画像は，使用用途に合わせて，高さや幅，フォーマット形式などに対する様々な加工が施されることが多い．しかし，コンテンツ指向アドレッシングを採用する IPFS では，これらの加工された派生画像がオリジナルとは異なるデータとして管理されるため，派生画像の再活用が難しい．その結果，同じ加工処理を何度も繰り返したり，多数の派生画像がストレージ容量を圧迫したりする不具合が発生する．本研究では，動的画像変換と派生画像の効率的な検索機構を IPFS へ実装し，派生画像の再活用を可能にする．また．構築した IPFS システムを用いた実験の結果から，本提案機構が画像処理のオーバーヘッドを削減できることを示す．

キーワード　画像処理，ストレージ管理，NFT，IPFS

1　はじめに

1.1　デジタルアートの流通

近年，デジタルアートなどのデジタルコンテンツを安全に流通するための仕組みとして，Non-Fungible Token（以下，NFT）が活用されている．NFT を活用しているマーケットの例として，OpenSea[1]や Magic Eden[2]，LooksRare[3]などがある．NonFungible が行った調査 [1] によると，NFT を利用して 1 年間でやり取りされている金額（US ドル）は，図 1 のようになる．2019 年では，約 2400 万ドル，2020 年では，約 8200 万ドル，2021 年には約 170 億ドルであり，ここ数年で急成長していることがわかる．

NFT は，ブロックチェーンをベースとした暗号資産で，同じトークンは存在せず，それぞれのトークンを識別するのに適しているという特徴がある [2]．そのため，NFT では，イラストやアート作品，写真などのデジタル画像の他，音楽作品や，ゲームアイテム，イベントチケットなどがやり取りされている．

ブロックチェーンは，ブロックサイズが大きくなるとコンセンサスにかかる時間が長くなるという特徴がある．加えて，デジタル画像はファイルサイズが大きくなりやすい．この課題を回避するため，いくつかの主要な NFT マーケットでは，コンテンツのメタデータやスマートコントラクトのみをチェーン上に保存し，コンテンツはオフチェーンに保存している．このコンテンツを保存する外部ストレージの 1 つとして，InterPlanetary File System（以下，IPFS）が使用されている．

IPFS は Protocol Labs が開発したプロトコルとネットワーク実装であり，Peer-to-Peer（以下，P2P）をベースとした非集中的な分散ファイルシステムである [3]．IPFS は，コンテンツの管理にコンテンツ指 S 向アドレッシングという方法を用いている．従来のファイルシステムや HTTP プロトコルの様に，ファイルが保存されている場所にもとづいてアクセスするロケーション指向アドレッシングとは異なり，コンテンツのデータにもとづいた ID でアクセスする．この ID は，コンテンツ ID（以下，CID）と呼ばれている．

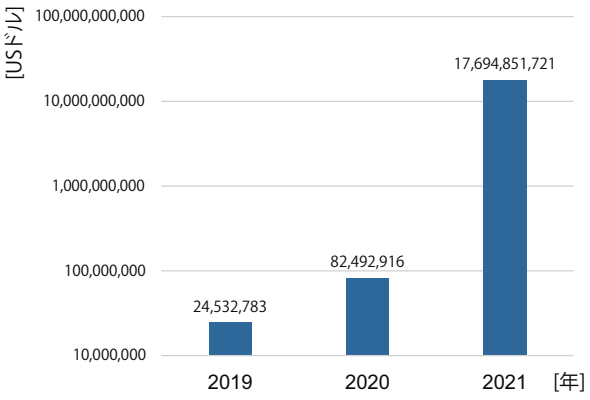

図 1　NFT でやり取りされた金額

CID はデータにもとづいて生成されるため，コンテンツのデータが変更されると CID も変化するという特徴がある．これにより，IPFS はコンテンツが改ざんされたかどうかを検証することができ，P2P でも信頼したデータの送受信が可能となる．

1.2　IPFS での派生画像管理の課題

IPFS で画像を扱う際の課題として，派生画像の管理に関する問題がある．派生画像とは，あるデジタル画像（以下，オリジナル画像）に対して，画像の編集や加工などを行い生成された画像群のことを指す．派生画像の具体例を図 2 に示す．

1：https://opensea.io/
2：https://magiceden.io/
3：https://looksrare.org/

図 2　派生画像の具体例

NFT で売買された画像はそのまま使用されるだけではなく，SNS のアイコンや Web ページへの埋め込みなど様々な用途で使用される．その際，必要に応じて画像サイズの変更やトリミング，色加工処理などが行われる．このように同じ画像から派生した，データが異なる画像群が生成され利用されている．IPFS 上では，これらの派生画像は画像編集や加工によりデータが変更されているため，異なる CID で管理される．また，CID はハッシュ関数を用いて生成されたランダムな文字列であり，CID から画像同士の派生関係を推測するのは困難である．そこで本研究では，これらの同一画像からの派生画像群を効率的に扱う方法について提案，実装，実験による検証を行う．

1.3　提　　案

IPFS 上での派生画像の再利用を促進するために，以下の 3 つの機能を実現する．初めに，IPFS 上での派生画像の作成を容易にするために，画像の動的変換の機能を実現する．次に，同じユーザによる派生画像の再利用を効率的に行うために，オリジナル画像の CID と行った編集内容を示すパラメータの組み合わせと，変換後の画像の CID の関連性を保存する．最後に，様々なユーザによる派生画像の再利用を可能にするために，ピア間で変換前後の CID 同士の関連性の共有を可能にする．

2　関 連 研 究

2.1　IPFS のコンテンツに関する研究

IPFS に関する様々な先行研究のうち，本節では IPFS のコンテンツに着目した研究に関して説明する．サポートベクター回帰を用いて，IPFS のコンテンツの複製数を動的に管理する Dongsheng らの研究 [4] がある．過去のコンテンツへのアクセス頻度から数時間後のアクセス数を予測し，予測結果に応じて複製数を増減させることで，コンテンツのアクセス待ち時間を効果的に短縮した．本研究とは IPFS コンテンツを動的に管理するという点で類似しているが，Dongsheng らの研究では単一コンテンツの複製数管理に着目しているのに対し，本研究ではある画像から派生した複数コンテンツの管理に着目している点が異なる．そのため，提案システムと組み合わせることで，利用頻度の高い派生画像の複製数を増加させることで待ち時間を短縮することが可能であると考える．

他にも，DNSLink と分散型の識別子（DID）を組み合わせて，IPFS 上で自己検証することができる，データが更新可能なコンテンツを実現した Nikos らの研究 [5] がある．コンテンツに検証可能な証明書を付与することで，取得したコンテンツがリクエストしたものであるかどうかを検証できるようにした．本研究とは，データ内容が変更されるコンテンツを対象としている点で類似しているが，Nikos らの研究では変更されるコンテンツの検証に着目しているのに対し，本研究では内容が変更されるコンテンツの生成や，アクセスの効率化について着目している点が異なる．

また，EU 一般データ保護規則の忘れられる権利を満たすために，IPFS にコンテンツ削除のプロトコルを実装した Politou らの研究 [6] がある．コンテンツの作成者やコンテンツを委任された人などの，権利を持っている人のみがコンテンツ削除の依頼をできるように，コンテンツに所有権の情報を付与し，削除リクエストの有効性を検証できるようにした．本研究とはコンテンツに着目している点で類似しているが，Politou らの研究ではコンテンツの削除に着目しているのに対し，本研究ではコンテンツの作成や管理に着目している点が異なる．

2.2　IPFS の性能に関する研究

IPFS の設計と実装について解説し，パフォーマンス評価を行った Trautwein らの研究 [7] がある．この研究では，IPFS が 152 カ国，2700 以上の自律型システムで使用されていることを明らかにした．また，コンテンツの取得と配信のベンチマーキングを行うために，異なる地域に設定した 6 つの仮想マシンを使用した．設定された地域は，バーレーン・シドニー・ケープタウン・カリフォルニア・フランクフルト・サンパウロである．これらの仮想マシンをパブリックな IPFS ネットワークに接続し，データを収集・分析した．その結果，IPFS のコンテンツの公開と取得にかかる遅延が，幅広いユースケースにおいて許容できることを示した．

3　実　　装

Go で実装された IPFS 標準実装である Kubo[4] を拡張して，1.3 節で提案した機能を実装した．使用した Go のバージョンは，go1.17.9 である．また，使用した Kubo のバージョンは，v0.12.2 である．加えて，ノード間での変換前後の CID の関連性の共有は，go-bitswap[5] のモジュールに実装した．つまり，提案システムでは，近隣のピアのみと関連性の共有が可能である．

提案システムが対応可能な画像処理は，リサイズとトリミングのみに限定した．リサイズは，高解像度のオリジナル画像を，Web 上に埋め込む際に軽量化するために使用されると考え，採用した．また，トリミングは，SNS のアイコンやバナーなどに NFT のイラストを設定する際に使用する頻度が高いと考えたため，採用した．

3.1　動的画像変換

初めに，IPFS 上での派生画像の生成を容易にするために，パラメータを用いた動的画像変換を実現する（図 3）．オリジナル画像の CID とどのような編集を行うかを示すパラメータ

4：https://github.com/ipfs/kubo
5：https://github.com/ipfs/go-bitswap

表 1　画像のトリミングを指示するパラメータ

パラメータ	説明
x	トリミング後の画像（左上）の x 座標
y	トリミング後の画像（左上）の y 座標
w	トリミング後の画像の幅
h	トリミング後の画像の高さ

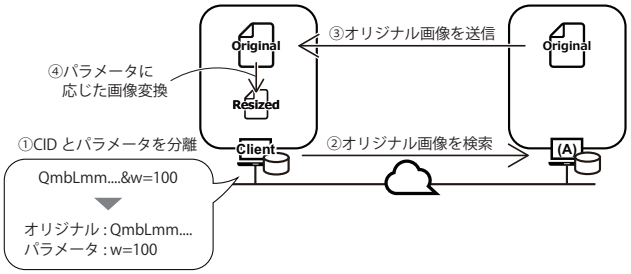

図 3　画像の動的変換の流れ

を組み合わせることで，IPFS 上でどのような派生画像の作成を行うかの指示を可能にする．CID とパラメータを，CID で使われることのない文字である “&” で接続することにより，CID とパラメータの判別を行う．トリミングを行う際のパラメータの例を，表 1 に示す．例えば，オリジナル画像の左上を原点として，右に 100px, 下に 200px の点から，幅 300px，高さ 400px でトリミングを行う場合，提案システムでは $< CID > \&x = 100, y = 200, w = 300, h = 400$ と指定する．

パラメータに応じた画像の動的変換を行うために，IPFS 上でパラメータを付与した CID を扱えるように，go-cid[6]の実装を変更した．変更後の CID 構造体は，以下のようになる．与えられた入力に “&” の文字が含まれているかどうかを確認し，含まれていれば “&” を目印として CID とパラメータを分離する．オリジナル画像の CID は str 変数に，パラメータは param 変数にそれぞれ格納する．

```
type Cid struct {
    str   string
    param string
}
```

続いて，パラメータに応じた動的画像変換について説明する．まず，オリジナル画像を CID 構造体の str 変数のみを用いて検索する．オリジナル画像の取得後，param 変数を参照して画像変換を行い，変換後の画像を取得する．取得した変換後の画像を IPFS のデータストアに追加することによって，変換後の画像の CID を取得する．以上の流れで，パラメータを用いた IPFS 上での動的画像変換を実現した．

画像変換に関して，オリジナル画像を取得後に検索を行ったノードで実施する方法と，検索対象となっているコンテンツを所持している端末で実施し，変換後の画像を送信する方法がある．前者の場合，オリジナル画像が検索した端末に複製されるため，使用頻度の高いコンテンツが様々なピアに分散され，取得が容易になる．また，画像処理をそれぞれのピアが実施するため，画像処理の負荷分散が期待できる．一方で，行われる画像処理がデータサイズが縮小する処理だと仮定した場合，変換後の画像が送信できる後者と比較して，ネットワーク上のデータ転送量が増加する．後者の場合，オリジナル画像を自身のデータストアのみに限定することができる．また，前述の仮定を考慮すると，ネットワーク上のデータ転送量を削減することができる．一方で，オリジナル画像を所有している端末に画像処理の負荷が集中する．画像変換を行うメリット・デメリットは対照的になっている．そのため，今回の実装では IPFS の分散性を重視し，クライアント上で画像処理を行うように実装した．

6：https://github.com/ipfs/go-cid

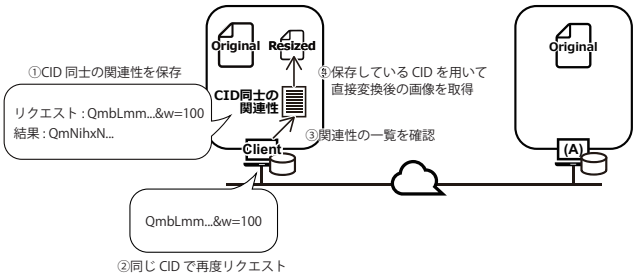

図 4　関連性の保存と再利用

3.2　関連性の保存と再利用

続いて，上記で作成した変換後の画像を再利用するために，変換前後の CID 同士の関連性を保存する仕組み（図 4）について説明する．オリジナル画像の CID と画像変換に関するパラメータの組み合わせと，変換後画像の CID の関連性の保存は，動的変換で取得した変換後の画像を IPFS に追加した後に行う．パラメータ付きの CID と変換後の CID の組み合わせを一対一で保存する．現在の実装ではテキスト形式のファイルとして，クライアントのストレージに保存している．また，保存した CID 同士の関連性を利用するのは，CID とパラメータを分離した後に行う．保存しているファイルから，CID 同士の関連性を 1 行ずつ読み出し，リクエストされたパラメータ付きの CID が存在するかを確認する．存在していた場合は，変換後の画像の CID を利用してコンテンツを検索する．存在していない場合は，オリジナル画像を検索して動的変換を行う．

3.3　ピアへの関連性の共有

前述した CID 同士の関連性は，端末内のストレージに保存されているため，別の IPFS ノードから利用することができない．そのため，他のピアが保存している CID 同士の関連性をピアからクライアントに伝達する方法を実装した（図 5）．ピア間で CID 同士の関連性の共有を可能にするために，go-bitswap のメッセージを拡張した．拡張後のメッセージ構造体は以下のようになる．

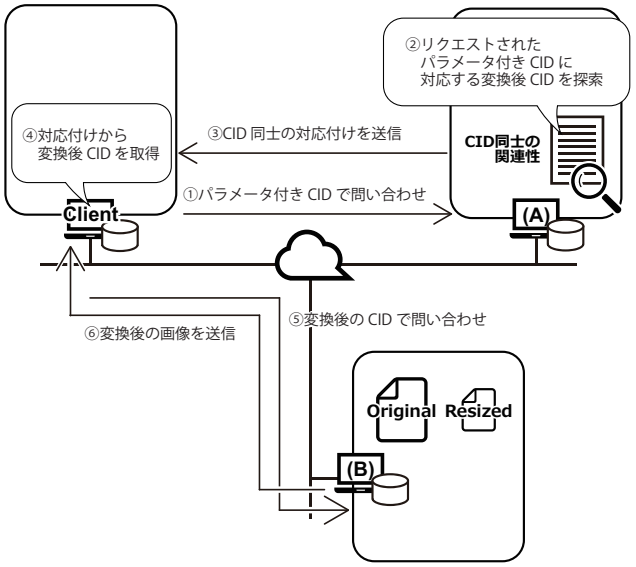

図 5　CID 同士の関連性を他のピアへ共有

```
type impl struct {
    full            bool
    wantlist        map[cid.Cid]*Entry
    blocks          map[cid.Cid]blocks.Block
    blockPresences map[cid.Cid]pb.Message_Block
                       PresenceType
    corresponds     map[cid.Cid]cid.Cid
    pendingBytes    int32
}
```

CID 同士の関連性をプログラム上で実現するために，キーを変換前のパラメータ付きの CID として，バリューを変換後の CID としたマップ変数に保存した．関連性を共有する具体的な流れとしては，ピアからパラメータ付きの CID でブロックの問い合わせがあった場合，保存している関連性を検索する．関連性が見つかった場合には，マップ変数に変換前後の CID の関連性を代入して返信する．返信を受け取った IPFS ノードは，このマップ変数を確認してリクエストに対応した変換後の CID を得る．取得した変換後の CID を利用してコンテンツを検索することで，ピアに保存された CID 同士の関連性を活用することができる．

3.4　提案システムの全体像

以上の実装を踏まえて，提案システムで IPFS のコンテンツを取得する流れを図 6 に示す．まず，クライアントに変換前後の CID 同士の関連性があるかどうかで分岐する．関連性がある場合には，変換後の CID を用いたコンテンツの探索とオリジナル画像の動的変換を同時に実行し，早く取得できたものを結果として使用する．関連性がない場合には，ピアへの関連性の問い合わせとオリジナル画像の動的変換を同時に実行する．2 種類の方法で同時に検索を行う理由は，対象となる画像や変換によって最適な検索方法が異なるため，どのような場合でも最適な方法で取得を可能にするためである．どちらの場合でも，いずれかの方法でコンテンツを取得できた場合には変換後の画像を利用し，取得できなかった場合にはコンテンツが取得できないというエラーを表示する．

また，提案システムのコンテンツ取得方法は，コンテンツをどこから取得するかと，どのように取得するかによって，大まかに 5 つに分類することができる．取得場所に関してはクライアントのローカル・データストアとピアのデータストアの 2 種類である．どのように取得するかは，オリジナル画像を取得して変形する方法と，変換前後の CID 同士の関連性を利用して変換後の画像を直接取得する方法である．また，ネットワーク上にコンテンツが存在しないなどの理由で取得できない場合もある．詳しい取得方法をまとめたものを以下に示す．

（1） クライアントが所持している変換後の画像を取得
（2） ピアが所持している変換後の画像を取得
（3） クライアントが所持しているオリジナル画像を
　　取得して変形
（4） ピアが所持しているオリジナル画像を取得して変形
（5） 取得不可能

提案システムがどのように変換後の派生画像を取得するかは，クライアントとピアの状態に依存する．クライアントとピアの状態として，オリジナル画像を所持しているか，変換後の画像を所持しているか，変換前後の CID の関連性を知っているかの 3 つがある．それぞれの状態において，提案システムがコンテンツを取得する方法を，図 7 に示す．ノードの状態によっては，複数種類の方法で取得できる可能性があるが，その場合は上記の箇条書きの数字が最も小さいものを図中に記載した．

4　評 価 実 験

4.1　概　　要

変換前後の CID 同士の関連性の保存・共有が，動的画像変換のオーバーヘッドを削減できているか検証するために，評価実験を行った．ネットワークの状況などが及ぼす影響を確認するために，異なる条件を持つ複数ノードが接続しているプライベートな IPFS ネットワークを構築した．実験環境を図 8 に示す．クライアントはコンテンツのリクエストを行うノードで，ノード A，B，C は変換前後の画像の保持や，クライアントへ変換前後の CID 同士の関連性を伝達するノードである．クライアントとノード B の間には 100ms のレイテンシを、ノード C との間には 200ms のレイテンシをそれぞれ設定した．レイテンシは，tc コマンドを用いて設定した．ノードそれぞれの CPU，メモリ，OS を，表 2 に示す．

また，実験の対象とする編集は 3 種類であり，それぞれ画像の縮小と拡大，そしてトリミングを行うものである．縮小は，NFT でやり取りされているような高解像度の画像 [8] を Web で掲載するために縮小することを想定して設定した．また，拡大はデジタル画像に対して本来あまり行われない処理であるが，2 種類の方法で同時にコンテンツを検索する機能が有効であるかを確かめるために計測した．さらに，トリミングは編集内容による実行時間の差を計測するために実験を行った．縮小の場

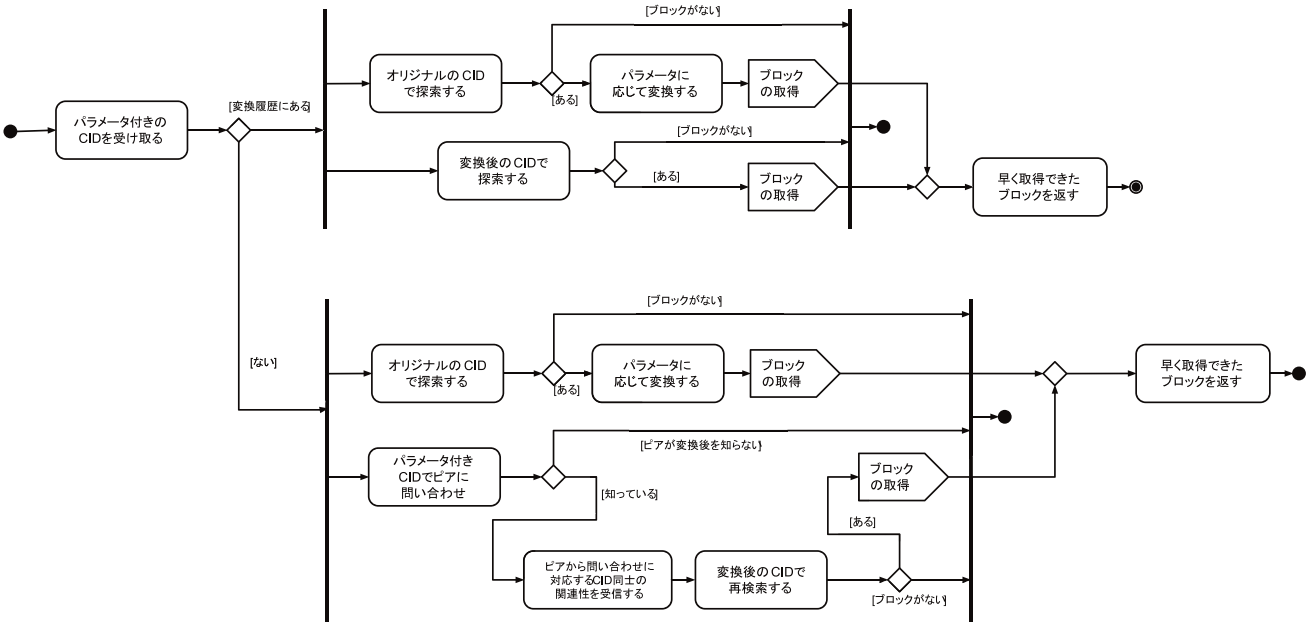

図 6　提案システムの流れ

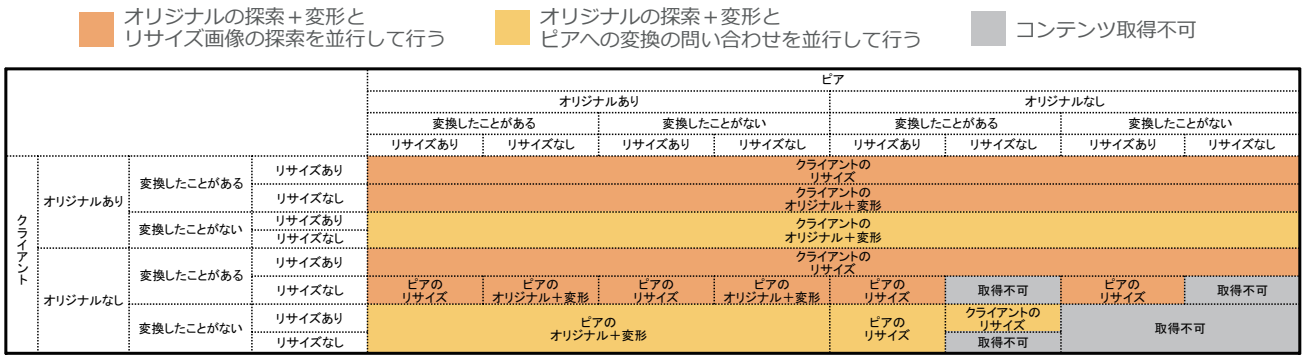

図 7　提案システムの状態

表 2　ノードの性能

	クライアント	A	B	C
CPU	AMD® Ryzen 9 3900x	Intel® Core™ i3-6100U	Intel® Core™ i7-10700U	Intel® Core™ i7-10700U
メモリ	16GiB	12GiB	32GiB	32GiB
OS	Ubuntu 22.04.1 LTS	Ubuntu 20.04.5 LTS	Ubuntu 20.04.5 LTS	Ubuntu 20.04.5 LTS

合と比較しやすくするため，同一のオリジナル画像を使用し，変換後の画像の幅と高さも同じ値に設定した．これにより，編集の内容がコンテンツの取得時間に及ぼす影響の調査も行った．それぞれの画像の変換前と変換後の詳細は，表 3，4，5 のようになっている．

コンテンツの取得方法として，以下の 10 種類の方法を計測した．それぞれの取得方法を 10 回ずつ実行し，平均取得時間を算出した．

表 3　拡大する画像の詳細

	オリジナル画像	変形後
ファイルサイズ	369KB	2.09MB
ファイルフォーマット	PNG	PNG
幅	640px	2048px
高さ	640px	2048px

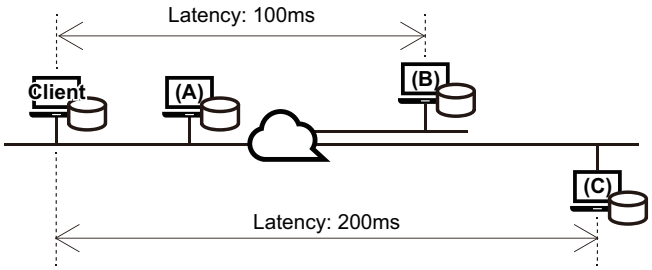

図 8　実 験 環 境

表 4　縮小する画像の詳細

	オリジナル画像	変形後
ファイルサイズ	2.66MB	244KB
ファイル形式	PNG	PNG
幅	2048px	640px
高さ	2048px	640px

表 5　トリミングする画像の詳細

	オリジナル画像	変形後
ファイルサイズ	2.66MB	228KB
ファイル形式	PNG	PNG
幅	2048px	640px
高さ	2048px	640px

(A)　クライアントからオリジナル画像を取得して動的変換
(B)　ノード A からオリジナル画像を取得して動的変換
(C)　ノード B からオリジナル画像を取得して動的変換
(D)　ノード C からオリジナル画像を取得して動的変換
(E)　クライアントから CID 同士の関連性を取得し，A から変換後の画像を取得
(F)　クライアントから CID 同士の関連性を取得し，B から変換後の画像を取得
(G)　クライアントから CID 同士の関連性を取得し，C から変換後の画像を取得
(H)　ノード A から CID 同士の関連性を取得し，B から変換後の画像を取得
(I)　ノード B から CID 同士の関連性を取得し，B から変換後の画像を取得
(J)　ノード C から CID 同士の関連性を取得し，B から変換後の画像を取得

4.2　結　　果

実験結果を図 9，10，11 に示す．まず，拡大の場合の結果について動的変換を行う条件 (A)〜(D) では，取得時間がノード間のレイテンシに応じて増加している．クライアントが保存している CID 同士の関連性を利用して，変換後のコンテンツを取得する条件 (E)〜(G) でも，取得時間はノード間のレイテンシに応じて増加している．特に，条件 (G) での実行時間が条件 (C) よりもわずかに長いため，オリジナル画像を取得して動的変換を行うものが早く終了した結果であると考える．ピアから変換前後の CID を伝達される条件 (H)〜(J) では，ノード間のレイテンシに応じた実行時間の増加が見られない．これも，オリジナル画像を取得して動的変換を行うものが早く終了した結果であると考える．

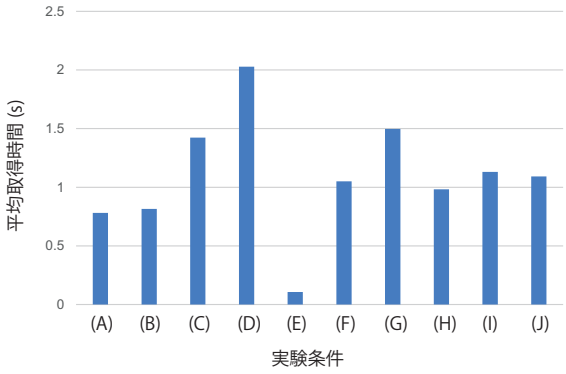

図 9　画像を拡大する場合の平均取得時間

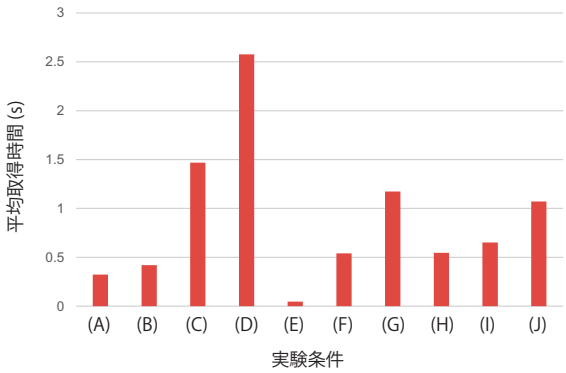

図 10　画像を縮小する場合の平均取得時間

縮小，トリミングに関して，グラフの概形は拡大と一部を除き同様である．異なる点としては，条件 (G)，条件 (H)〜(J) で変換後の CID を用いてアクセスする方が動的変換より早く終了していることがある．そのため，実行結果がノード間のレイテンシに応じて増加している．

3 種類の実験結果をまとめたものが図 12 である．まず，画像の動的変換について，近くのノードからオリジナル画像を取得する条件 (A)，(B) では拡大の方が縮小やトリミングに比べて長くなっている．一方で，遠くのノードから取得する条件 (C)，(D) では，拡大よりも縮小やトリミングの実行時間が長くなっている。これは，縮小のオリジナル画像のデータサイズが拡大よりも大きいため，ノードの距離が遠くなると通信にかかる時間も増加していることが原因だと考える．リサイズ後の画像を取得する条件 (E)〜(J) では，拡大の変換後の画像のデータサイズが大きいため，どの場合でも実行時間が一番長くなっている．

4.3　考　　察

いくつかのネットワーク条件，編集内容で実験を行った結果，大半の場合でクライアントやピアが保持している CID 同士の関連性を利用した場合の実行時間が短いことがわかった．特に，クライアントが履歴を保持しており，かつ近くのノードに変換

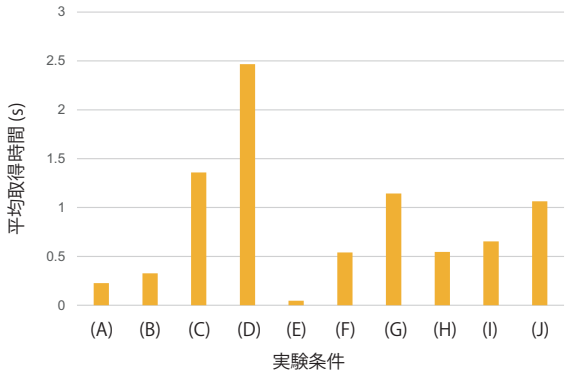

図 11　画像をトリミングする場合の平均取得時間

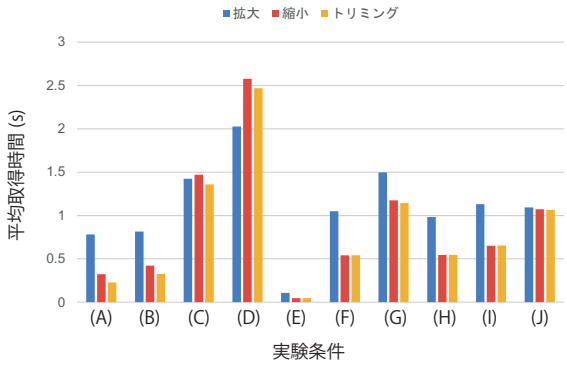

図 12　平均取得時間の比較

後の画像が存在する場合（条件 (E)），高速に変換後の画像を取得することが可能であった．一方で，行われる編集の内容や変換後の画像が保存されているノードとの距離によっては，動的変換を行った方が早く取得できる場合があることがわかった．例えば，遠くのノードがオリジナル画像よりもデータ量の多い変換後画像を保持している場合（条件 (G)）などである．他にも，画像変換を行うクライアントの PC 性能を変化させると，動的変換にかかる時間も変化するため，また異なった結果になることが考えられる．様々な条件によって早く取得できる方法が異なるため，変換後のコンテンツの探索とオリジナル画像の探索と動的変換を同時に実行することが取得時間の短縮に効果的であることがわかる．

5　おわりに

派生画像の再利用を促進するために，以下の機能を IPFS の標準実装である Kubo に実装した．初めに，IPFS 上で派生画像の作成を容易にするために，画像の動的変換の機能を実現した．CID の末尾に画像編集に関するパラメータを付与できるように実装を変更した．CID が有効かどうかを確認する前に分離し，オリジナルのコンテンツを取得した後にパラメータの値に応じた画像処理を行い，IPFS のデータストアに追加することで変換後の画像の CID を取得した．次に，同じユーザからの再利用を効率的にするために，オリジナル画像の CID と行った編集内容と，変換後の画像の CID の関連性の保存を実装した．ノードに変換前後の情報を保存しておき，オリジナルのコンテンツを探索する前に，変換前後の CID 同士の関連性を参照し変換後の CID を入手することで，画像処理にかかる時間を削減することができた．最後に，変換前後の CID 同士の関連性をピアと共有することで様々なユーザからの再利用を可能にするために，ピア間で対応の共有を可能にした．ピア間でやり取りされているメッセージを改変し，パラメータ付きの CID と対応する変換後の画像の CID を伝達できるようにした．評価実験の結果から，変換前後の CID の関連性の保存・共有は，画像のデータサイズやネットワークの状況によらず，効果的であることが示された．このことから，提案システムは，NFT のオフチェーンデータとして IPFS 上に保存された画像の活用や，作成した派生画像の再利用に貢献できる．

5.1　今後の展望

今後の展望として，まず，実施することのできる編集・加工の種類の増加がある．現状，パラメータとして付与できる編集の種類は，画像の幅・高さを指定して拡大・縮小を行う編集と，左上の座標と幅・高さを指定することでトリミングを行う編集の 2 種類である．今後，画像のトリミングや解像度の変更，画像の回転，色加工処理など，パラメータとして設定できる編集の種類を増やすことで，より多くのユーザのニーズに対応する必要がある．また，現状の提案システムは，ペイントソフトウェアなどを使用して編集・加工したデータが追加されても派生画像として活用することができない．これは，提案システムが，ペイントソフトウェアを通じて編集・加工された派生画像の編集内容を認識できないためである．ペイントソフトウェア上で編集履歴を管理する研究 [9] や，画像間の編集履歴を推測する研究 [10] を参考にすることで，パラメータでは指定することが難しいような編集・加工が施された派生画像を，提案システム上で関連付けて管理することが可能になると考える．

次に，複数の画像編集の組み合わせを実現することがある．現在の実装では，拡大・縮小かトリミングのどちらか一方しか適用することができない．だが，トリミング後の画像をリサイズするなど，複数の編集を組み合わせて使用する可能性がある．その機能を実現するために，複数画像編集の手順をどのように指定するかを検討する必要がある．複数画像編集の手順については，CID とパラメータの区切り文字として使用している “&” を複数設定することによって示す方法などを検討している．

最後に IPFS 上にある画像データから加工元が同じデータを見つけて統合することがある．適用できる編集の種類が増加することで，同じ編集をオリジナル画像に適用したとしても，最終的に完成する画像がデータが異なるものになる可能性がある．そのため，IPFS に保存されている画像データ群の中から同じ画像から編集・加工して作成された画像を収集し，オリジナル画像と適用された編集を特定することで，より多くの派生画像を活用することができる．同じオリジナル画像から作成された派生画像の特定には，重複画像の判定の研究（[11], [12]）を活用できると考える．この技術を活用することにより，すでに IPFS 上に保存されている派生画像の発見や，圧縮方式の違

いによって異なるデータになってしまった画像の集約が実現できる．

文　　献

[1] NonFungible.com. Yearly NFT Market Report 2021, 2021.

[2] Qin Wang, Rujia Li, Qi Wang, and Shiping Chen. Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges. 10 2021.

[3] Protocol Labs. IPFS Powers the Distributed Web. `https://ipfs.tech/`. (Accessed on 02/03/2023).

[4] Dongsheng Huang and Lijun Chen. Dynamic Replica Management based on SVR in IPFS. In *2019 IEEE 14th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pp. 591–597, November 2019.

[5] Nikos Fotiou, Vasilios A. Siris, and George C. Polyzos. Enabling self-verifiable mutable content items in IPFS using Decentralized Identifiers. *arXiv:2105.08395 [cs]*, 2021.

[6] Eugenia Politou, Efthimios Alepis, Maria Virvou, and Constantinos Patsakis. Implementing Content Erasure in IPFS. In *Privacy and Data Protection Challenges in the Distributed Era*. Springer International Publishing, Cham, 2022.

[7] Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. Design and evaluation of IPFS | Proceedings of the ACM SIGCOMM 2022 Conference. *SIGCOMM '22: Proceedings of the ACM SIGCOMM 2022 Conference*, pp. Pages 739–752, August 2022.

[8] NFTs Guru. Best NFT Art Size & Dimensions For Creators — Ultimate Guide. `https://nftsguru.com/make-nft/best-nft-art-size-dimensions/`, 5 2022. (Accessed on 02/13/2023).

[9] Hsiang-Ting Chen, Li-Yi Wei, and Chun-Fa Chang. Nonlinear revision control for images. *ACM Transactions on Graphics*, Vol. 30, No. 4, pp. 105:1–105:10, July 2011.

[10] Shi-Min Hu, Kun Xu, Li-Qian Ma, Bin Liu, Bi-Ye Jiang, and Jue Wang. Inverse image editing: recovering a semantic editing history from a before-and-after image pair. *ACM Transactions on Graphics*, Vol. 32, No. 6, pp. 194:1–194:11, November 2013.

[11] Ming Chen, Shupeng Wang, and Liang Tian. A High-precision Duplicate Image Deduplication Approach. *J. Comput.*, Vol. 8, pp. 2768–2775, 2013.

[12] Bin Wang, Zhiwei Li, Mingjing Li, and Wei-ying Ma. Large-Scale Duplicate Detection for Web Image Search. In *2006 IEEE International Conference on Multimedia and Expo*, pp. 353–356, 7 2006.