拡散レポジトリを用いたIPFS可用性向上技術

立命館大学
ワン　シュ，上山憲昭

研究背景

- IPFS
  - コンテンツアドレシングを用いた分散型p2pファイル保存や共有システム
  - Kademlia アルゴリズムに基づいた DHT (Distributed Hash Table)を利用して行われる

- 現行IPFSの課題
  - すべてのノードが同一のデータを重複して保持しない，各ノードがどのデータを保持するかは各ノードの自律的な判断で決まる
  - 常に個数以上のノードに保持させることを保証できない
  - 同一データが地理的に集中して保存される可能性

関連研究

- ピニング（Pinning）：特定のデータをピニングし，自身がそのデータを永続的に保存
- Filecoin：インセンティブを与え，ノードに長期間データを保持させること
  - ピニングを依頼することは煩雑、またピニングを行うノードの位置に応じた選択は困難
- クラスタ（Cluster）： 複数のIPFSノードを管理・調整する方法．データの分散保存や複製の作成を行い，耐障害性や可用性の向上
  - 設定が煩雑で、データが頻繁に更新される場合、ノード間でデータの整合性を保つことが困難

研究目的

- 障害や地震等の大規模災害に対する耐障害性の向上することや効率化が重要
- authority やデータ所有者が明示的に、データを永続的に保持するノードの数や、それらの位置を制御
- 効率的な IPFS のデータ配布方法
  - レポジトリを用いたデータ保存規則
  - NID付与アルゴリズム

提案手法(レポジトリを用いたデータ保存規則)

- レポジトリは特定のデータを永続的にピニング

- NID：今回提案された各レポジトリのバイナリID
- y ：NIDの最大ビット数
- k ：NIDの上位kビットとデータのハッシュ上位kビットを示す
- S_k：特定な上位kビットのノード集合に定義

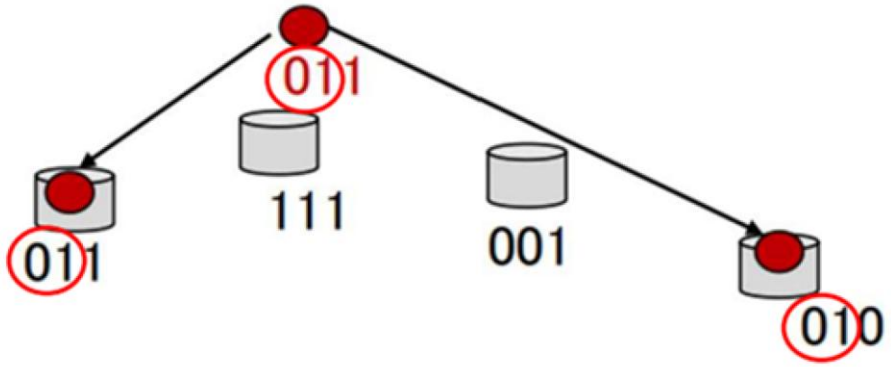


図1. y = 3, k = 2, S_k = (0,1)

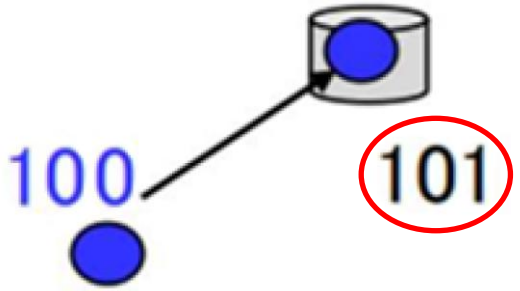


図2. y = 3, k = 3 S_k = (1,0,1)

提案手法(レポジトリを用いたIPFS可用性向上技術)

- 制御パラメータ k により、同一データを保持するレポジトリ数を調整可能
  - 小さい k → 高可用性(多重保持)
  - 大きい k → 低コスト(少数保持)
- 各ファイルの人気度により k の値を変更可能
- 小さな制御負荷で柔軟な可用性制御が可能

提案手法(NID付与アルゴリズム)

- 可能な限り空間的に離れた場所で同一なNIDを付与
- T_n がノードnから$S_{k+1}(S_k, 0)$と$S_{k+1}(S_k, 1)$へのホップ距離の総合と定義
- 貪欲法でT_nが最小化の第 k+1 ビットを割当てるアルゴリズムを確立

Algorithm 1 Peer ID Assignment Algorithm

Input: Graph $G = (V, E)$ with coordinates (x_v, y_v), max depth K, weight α
Output: K-bit Peer ID prefixes, for all repositories
for $k \leftarrow 0$ to $K - 1$ **do**
    Partition current set S_k into two subsets $S_{k+1}(0)$ and $S_{k+1}(1)$
    Compute cost $T_n = d_{hop}(n, S_{k+1}(0)) + d_{hop}(n, S_{k+1}(1))$
    Add geographic penalty:

$$C = \sum_{n \in S_k} T_n + \alpha \cdot \sum_{i,j \in S_{k+1}} d_{geo}(i, j) \quad (1)$$

    Choose partition minimizing C
    Assign $(k + 1)$-th bit = 0 or 1 to nodes in each subset
end for
Return all nodes with K-bit prefixes

提案手法(NID付与アルゴリズム)

- Authorityありの場合
  - 集中的にID割当 → 最短ホップ距離を最小化するアルゴリズム
- Authorityなしの場合
  - 近隣レポジトリの座標を取得
  - 空間的に最も離れたビット値を自律的に設定

- データの冗長化と空間的分散を両立、大規模障害への耐性を強化

性能評価条件

- Chunking (データ分割):
  - 大きなファイルを256KB単位の小さなチャンクに分割して管理
  - 各チャンクは一意のCID (Content Identifier) で識別される
  - ファイルを完全復元するには**全てのチャンク**を回収する必要があり、一つでも欠けるとデータが利用不能

- 公平性の確保
  - 従来方式と提案方式で同一数のノードが保持するように調整
  - ノード数，トポロジ構造，遅延パラメタは全て同一
  - 災害シナリオでは，災害範囲の中心を固定し，半径のみを変化させる

性能評価条件(一般シミュレーション)

- 災害シミュレーション
  - 固定点を中心とした半径 R の損害範囲を設定
  - 範囲内のノードは停止 ⇒ データ取得不能
  - 半径を段階的に拡大し，ファイルの完全復元成功率を測定
  - 従来手法:ランダムピニング

- 一般シミュレーション
  - 障害なしの状態で両方式を比較
  - 10,000回のデータ取得要求を実行
  - ファイルサイズ:lognormal 分布
  - 人気度:Zipf 分布で，異なる θ 値(0.6~0.9)でデータセットを生成
  - 評価指標(各データブロックに対して)
    - 平均復元遅延
    - 平均ホップ数
    - 4ホップ以内の復元成功率

性能評価（災害シミュレーション）

- 米国商用ISP@homeネットワークトポロジー
- ファイルサイズが右に裾の長い重尾分布に従う対数正規分布

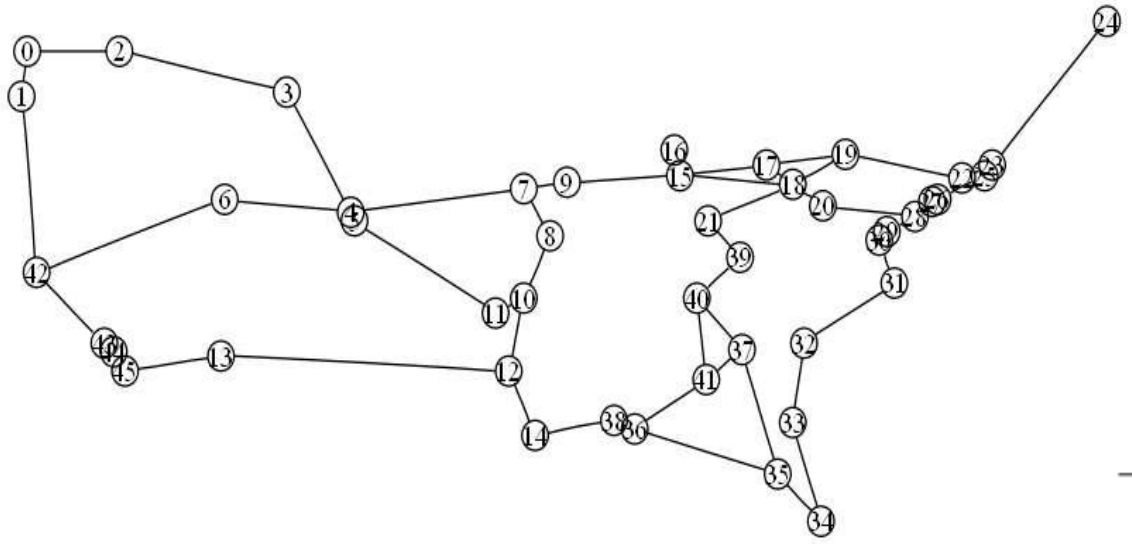


図3. ネットワークトポロジー

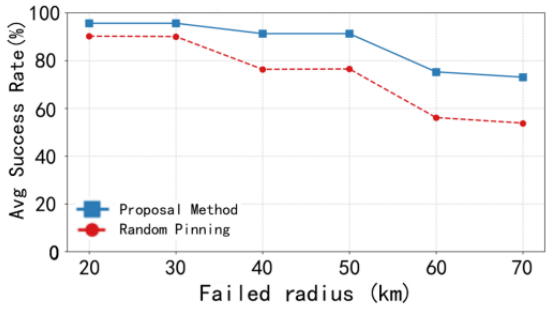
	MEAN	*MID*	*STD*	*MAX*	*MIN*
File Size	4.52	2.73	7.06	139.82	0.1

表1：1000 個ファイルの例（単位：Mb）

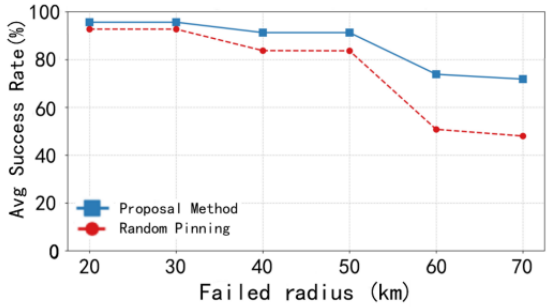
性能評価（災害シミュレーション）

- 災害シミュレーション結果
  - y = 5
  - 災害が拡大した場合提案方式は高い成功率を維持

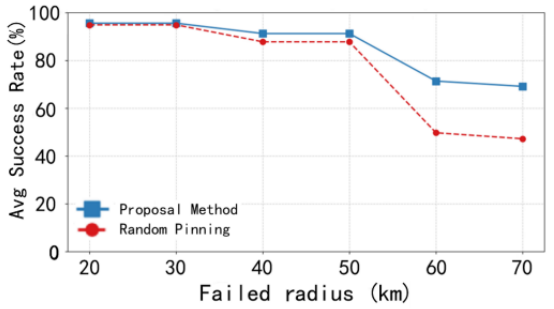
- ファイル数は 10, 100, 1000 の 3 パタンを評価し，それぞれの規模で両方式の性能を比較した



(a) 10 files



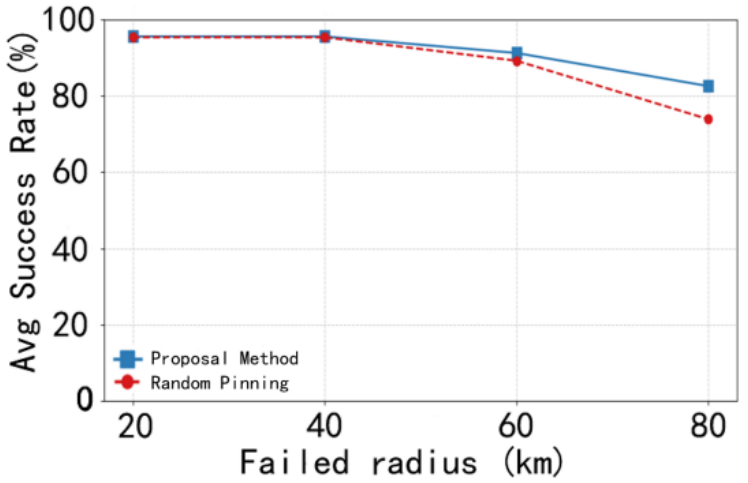
(b) 100 files



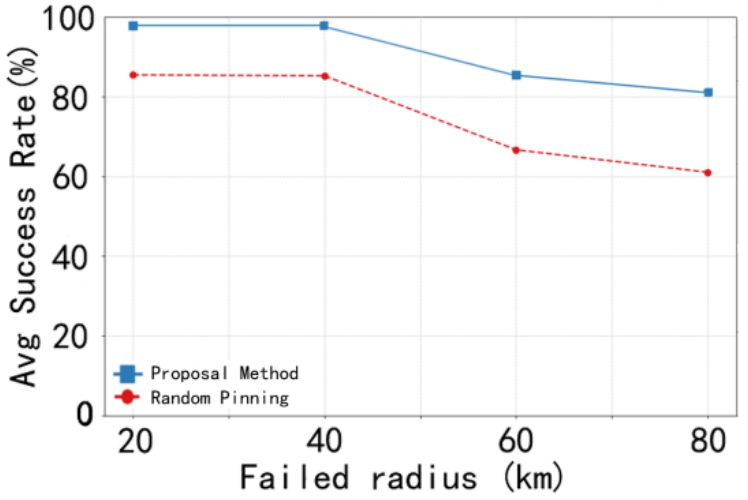
(c) 1000 files

性能評価（災害シミュレーション）

- 提案方式は高人気ファイルに対しては従来方式と同等の性能を維持
- 低人気ファイルにおいては復元成功率が高人気ファイルの場合を上回る
- 全体的にファイルの人気度が低いほど，従来方式に対して優位となる



(a) Many copies of data at all nodes

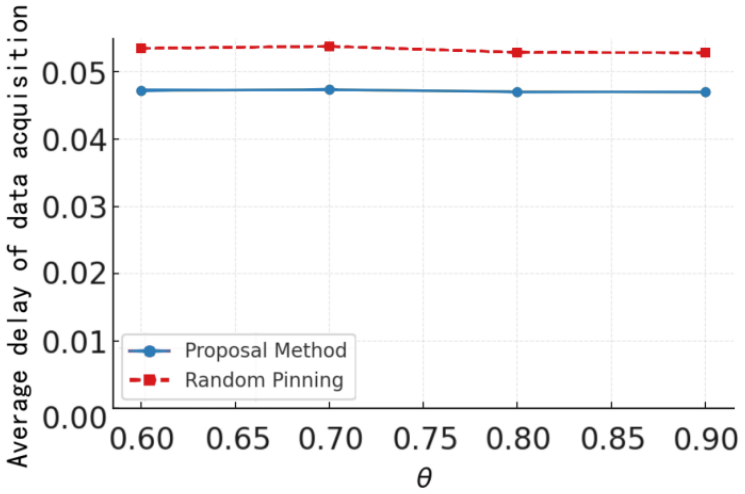


(b) Few copies of data at all nodes

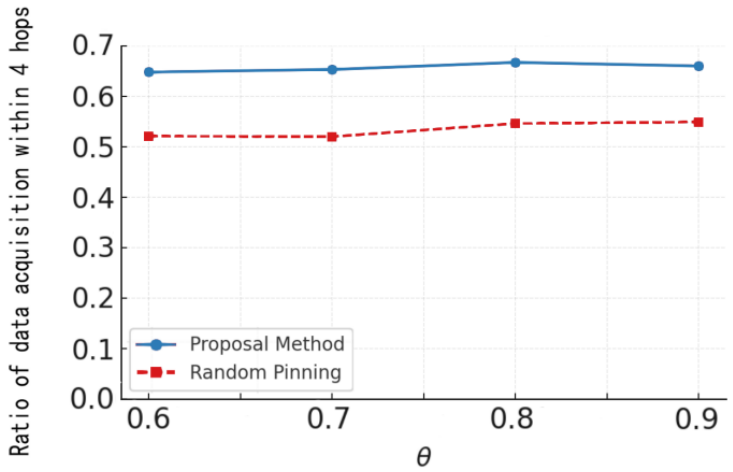
性能評価（一般シミュレーション）

- 一般シミュレーション結果

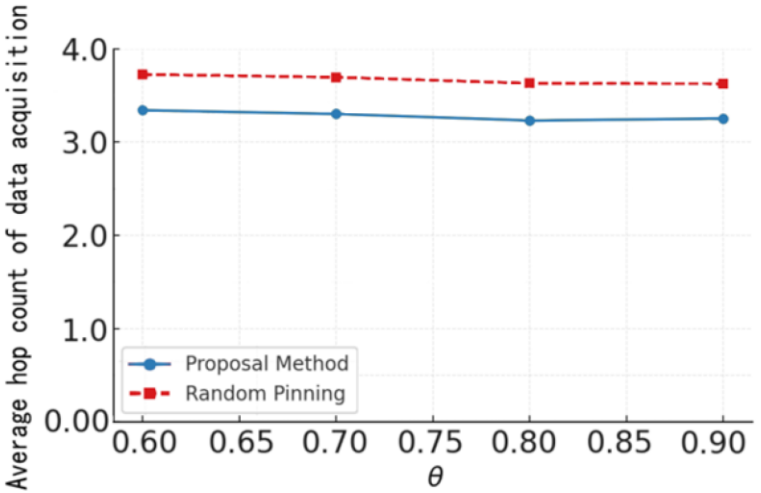
- 横軸が異なるθ値で:
  - 遅延 10%～25% 減少
  - 平均ホップ数 10%～25% 減少
  - 4ホップ以内成功率 10%～20% 向上



(a) Average delay of data acquisition



(c) Ratio of data acquisition within 4 hops



(b) Average hop count of data acquisition

まとめ

- 提案方式：レポジトリ導入＋NID付与アルゴリズム
- 大規模障害下でも高い復元成功率を維持
- 提案手法は従来手法に比べて，災害時・平常時の両方で有効性を確認
  - 特に低人気度ファイルに対しての効果が高い
- 今後の方針：
  - ネットワークトポロジーのサイズを増大
  - データ取得要求の数を増えて、異なる想定状態での表現を統計
  - 動的な状況を考慮