Osaka University

情報指向ネットワークへの適正と実現可能性を有するCLOCK-Proに基づいたキャッシュ置換方式の提案と評価

†大岡 睦，†オム スーヨン，‡阿多 信吾，†村田 正幸

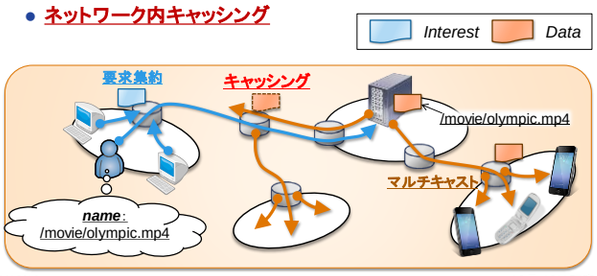
†大阪大学 大学院情報科学研究科
‡大阪市立大学 大学院工学研究科

発表内容

- 研究背景
  - ICN ルータにおけるキャッシング
  - キャッシュ置換方式の課題
- 提案手法
  - CUSH (CLOCK-Pro Using Switching Hash-tables)
    - ネットワークトラフィックに適した戦略
    - ルータで実現可能な低コストの実装
- 評価
  - 既存方式との性能比較
  - 実装コスト

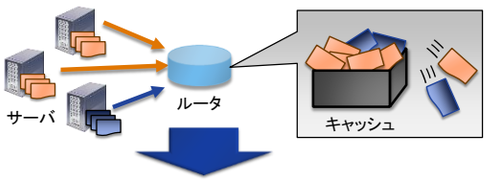
情報指向ネットワーク (ICN)

- ホスト中心ではなくコンテンツ中心
- コンテンツ拡散型の利用形態を支援
  - Interest 要求集約
  - Data マルチキャスト
  - **ネットワーク内キャッシング**



ICN におけるキャッシングの課題

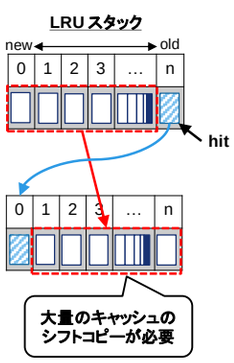
- 資源が限られた状況下でのキャッシング
  - 大量のコンテンツに対してルータのメモリ資源は有限



- **キャッシュ置換方式**
  - 人気のあるコンテンツのみをキャッシュ
- キャッシュ配置・判断方式
  - 複数のルータでキャッシュを分散

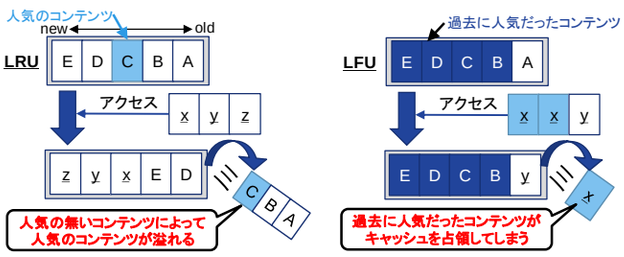
キャッシュ置換方式の実現課題

- ICN ルータハードウェアにおける実現性
  - LFU や人気度を計測する方式は過剰なメモリ・計算コストが必要
  - LRU でも高速アクセスが必須な環境では実現が困難
    - 例: LRU スタックの実装（右図）
- ネットワークトラフィックへの適正
  - ワンタイマーアクセス(scan)への耐性
  - チャンク単位アクセス(loop)への耐性
  - FIFO・Random・LRU などの単純な手法は耐性を持たない



ネットワークトラフィックの特性

- コンテンツの人気度に大きな偏りがある[1]
  - コンテンツ全体の約90%が1度しか要求されない
  - ワンタイマーコンテンツがトラフィックの40%を占める
- 既存の単純な方式はネットワークに不適



[1]F. Guillemin, et al, "Experimental analysis of caching efficiency for YouTube traffic in an ISP network," in Proceedings of the 25th International Teletraffic Congress, September 2013, pp. 1–9.

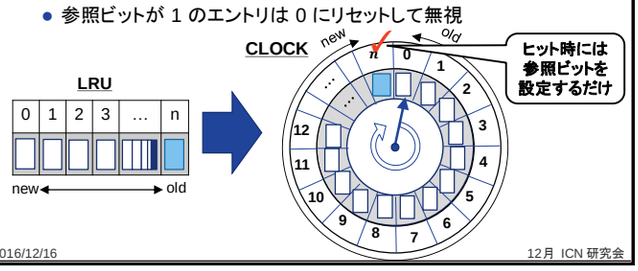
研究の目的とアプローチ

- 目的
  - ICN ルータにおけるキャッシュ置換方式の実現
    - ルータハードウェアでの実現可能な低コスト
    - ネットワークトラフィックへの適正
- アプローチ
  - LIRS/CLOCK-Pro [2,3] に注目
    - scan 耐性と loop 耐性を実現
    - キャッシュ履歴などの管理のための計算・メモリコストが課題
  - ルータにおいて実現可能な CUSH を提案・評価
    - CUSH: CLOCK-pro Using Switching Hash-tables
    - LIRS/CLOCK-Pro の長所を強化し、ネットワーク適性を実現
    - キャッシュ履歴による検索テーブルへのオーバーヘッドを削減

[2]F. N. Megiddo, et al., "ARC: a self-tuning, low overhead replacement cache," in Proceedings of the 3rd USENIX Conference on File and Storage Technologies, March 2004, pp. 115-130.
[3] S. Bansal and D.S. Modha, "CAR: Clock with adaptive replacement," in Proceedings of the 3rd USENIX Conference on File and Storage Technologies, pp.187–200, March 2004.

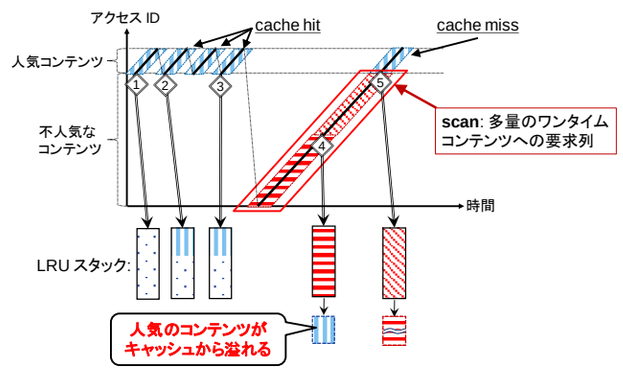
CLOCK の特徴

- 各エントリに参照ビットを設定
  - 各エントリあたり 1 bit だけのメモリオーバーヘッド
  - ヒット時には参照ビットを1に変更するだけの計算オーバーヘッド
- 時計の針のように円環バッファを巡回することで LRU を近似
  - 参照ビットが 0 のエントリを優先的に削除
  - 参照ビットが 1 のエントリは 0 にリセットして無視



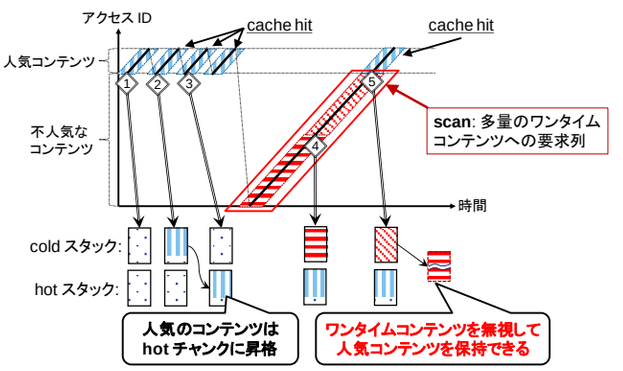
アクセスパターン scan による悪影響

- 多量のワンタイムコンテンツへの要求でキャッシュが溢れる



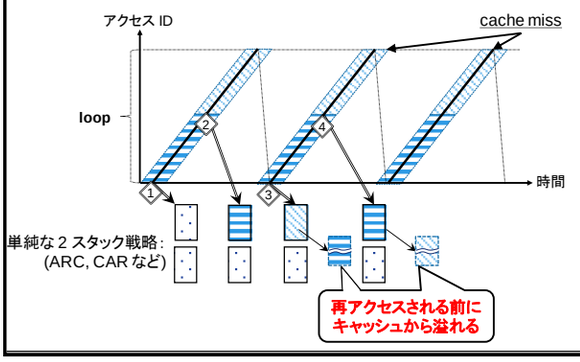
LIRS/CLOCK-Pro の scan 耐性

- コンテンツを hot (人気) と cold (不人気) の二種類に分類



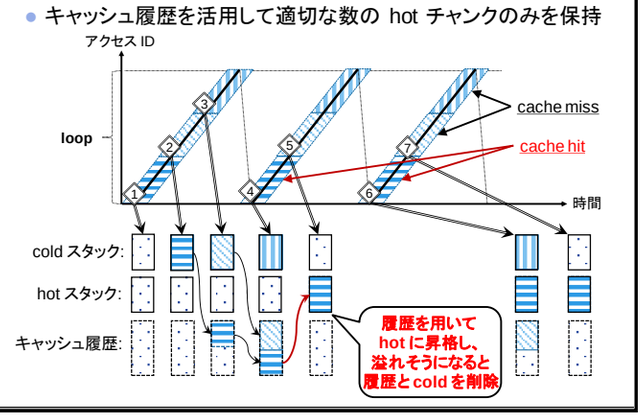
アクセスパターン loop による悪影響

- 毎回キャッシュ溢れが起きるようなアクセスの繰り返し
  - 大容量コンテンツへのチャンク単位アクセスによって発生



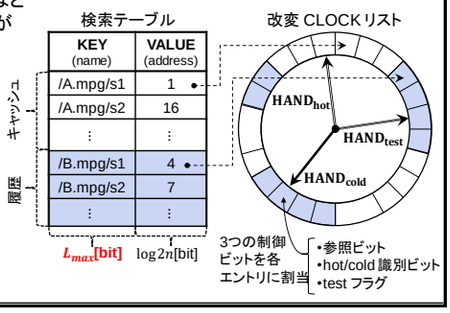
LIRS/CLOCK-Pro の loop 耐性

- キャッシュ履歴を活用して適切な数の hot チャンクのみを保持



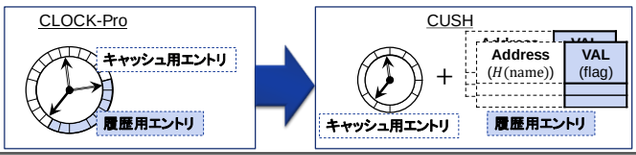
CLOCK-Pro のデータ構造と課題

- キャッシュ履歴数を超える長さの loop に対処できない
- キャッシュ履歴のオーバーヘッドが大きい
  - 検索テーブルで履歴用の name を保持する必要がある
  - 履歴数が多いほど CLOCK リストが肥大化し、巡回のための計算コストが大きくなる

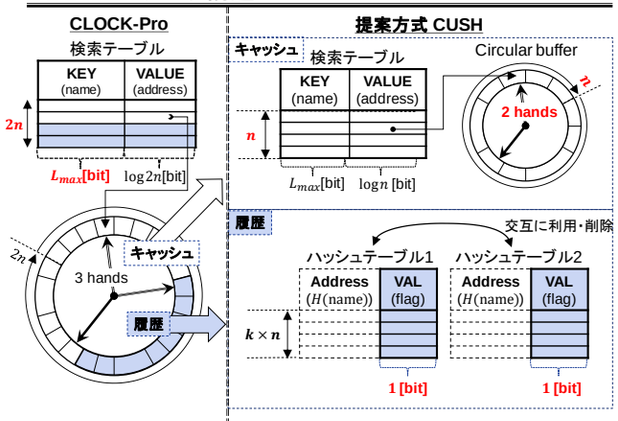


提案方式 CUSH

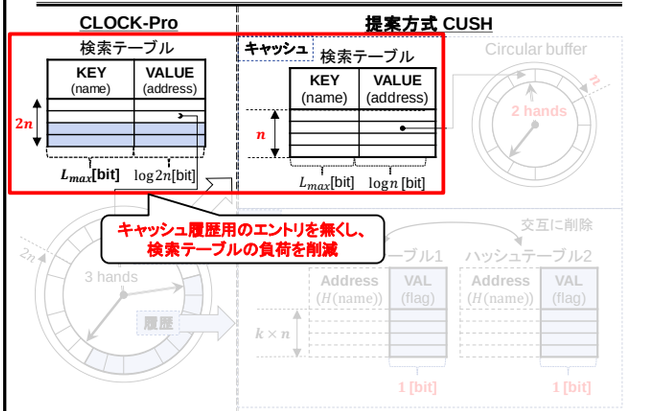
- ルータハードウェアでの実装を考慮した LIRS/CLOCK-Pro の近似アルゴリズム
  - CLOCK-Pro の特徴を引き継ぎ、scan 耐性を実現
  - loop 耐性を低オーバーヘッドに拡張可能
  - CLOCK-Pro のキャッシュ履歴のオーバーヘッドを削減
- キャッシュ用のエントリと履歴用のエントリを分離
  - 履歴用のエントリは衝突ありのハッシュテーブルとして保持
    - 過去にアクセスがあったか否かを1bitで管理
  - loop 耐性のために履歴数を増やしても巡回コストが増大しない



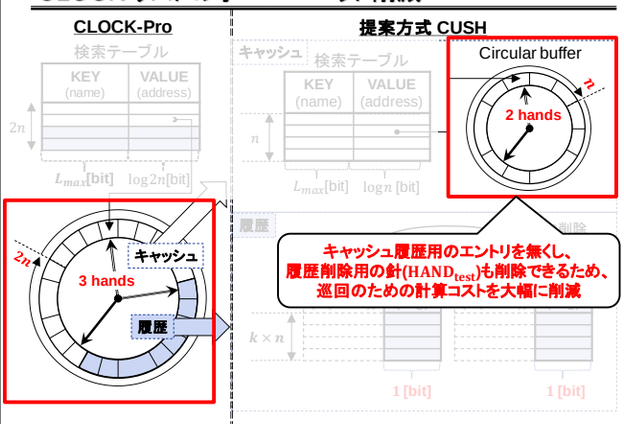
CUSH のデータ構造



検索テーブルのオーバーヘッド削減

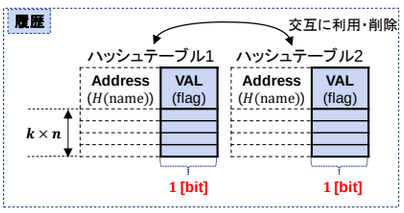


CLOCK リストのオーバーヘッド削減



ハッシュテーブルを用いたキャッシュ履歴の実装

- コンテンツ名のハッシュ値$H(name)$と対応するアドレスにキャッシュ履歴を保持
  - loop の繰り返しが検出できればよいので、1 bit の情報で十分
  - 検索・計算オーバーヘッドが小さく、拡張性が高い
- 2つのハッシュテーブルを交互に利用・削除
  - 履歴に登録されてすぐに履歴から削除されることを防ぐ
    - 人気コンテンツを hot チャンクに分類しやすくなる



評価

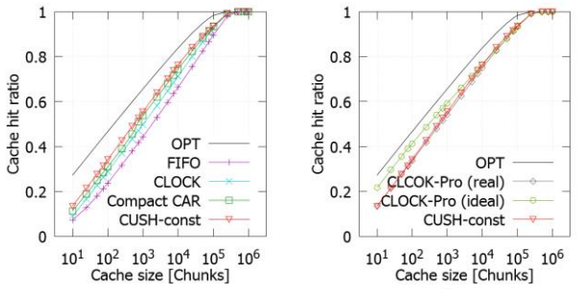
- ネットワークへの適性の評価
  - シミュレーションを用いたキャッシュヒット率評価
- ハードウェアでの実現可能性の評価
  - 空間計算量
    - キャッシュと履歴管理のための追加ビット数を評価
  - 時間計算量
    - 針の平均回転数に基いて評価

評価環境

- トポロジー：単一ノードトポロジー
- 比較方式
  - 最適手法： OPT
  - 単純な方式： FIFO・CLOCK
  - scan 耐性を持つ方式: Compact CAR
  - 提案方式と元となった方式： CUSH, CLOCK-Pro
    - キャッシュ履歴は衝突有りハッシュテーブルで実装
- トレースデータ
  - Zipf 則に従って人工的に生成したトレース
  - 阪大キャンパスの YouTube へのアクセスに基づくトレース
- キャッシュの設定： コンテンツ単位とチャンク単位
  - チャンク単位ではコンテンツをチャンクサイズで分割
    - 1500 Byte・15 KB・60 KB の3通り

人工的なトレース（コンテンツ単位）の場合

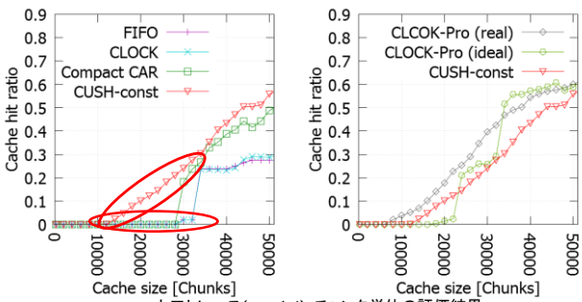
- CUSH は単純な方式と比較して3倍程度のキャッシュ効率
- CUSH が CLOCK-Pro と同等の性能を達成



人工トレース($\alpha = 1.0$):コンテンツ単位の評価結果
（左：単純な方式との比較、 右：CLOCK-Proとの比較）

人工的なトレース（チャンク単位）の場合

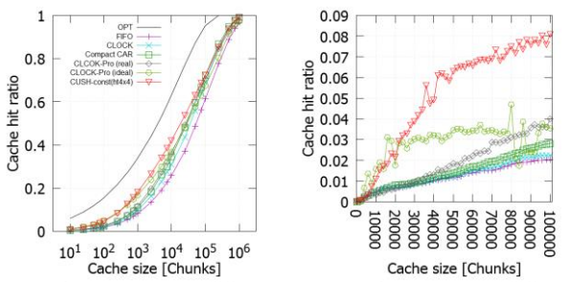
- キャッシュサイズが小さい状況でも loop をキャッシュ可能
  - 単純な方式はキャッシュサイズが大きくなるまでヒット率が0



人工トレース($\alpha = 1.4$):チャンク単位の評価結果
（左：単純な方式との比較、 右：CLOCK-Proとの比較）

単一ノードの評価結果：実トレースデータの場合

- 人工トレースとほぼ同じ結果
  - CUSH は人気度の頻繁な移り変わりに対応できるようにアルゴリズムに工夫を加えているため、CLOCK-Pro よりヒット率が高い



実トレース：コンテンツ単位　　実トレース：チャンク単位

キャッシュ置換方式の空間計算量の評価

- キャッシュサイズ n の場合のメモリオーバーヘッドを評価
  - CUSH は$k \times n$個の履歴を有すると仮定

各用途ごとのキャッシュ置換方式の空間計算量

キャッシュ置換方式	キャッシュの管理	履歴の管理	履歴用の検索テーブル
$\mathrm{LRU}_{\mathrm{DLL}}$	$O(n \log n)$	-	-
CLOCK	$\boldsymbol{O(n)}$	-	-
CLOCK-Pro	$\boldsymbol{O(n)}$	$\boldsymbol{O(n)}$	$\boldsymbol{O(n \log n)}$
CUSH	$\boldsymbol{O(n)}$	$\boldsymbol{O(k \times n)}$	-

- CLOCK と同等のキャッシュ管理コスト
- 履歴用の検索テーブルへのコストを削除

キャッシュミス時の時間計算量の評価

- 針の平均回転数に基いて時間計算量を評価
  - 針の回転数は1にセットされた参照ビットの数に応じて増えるため、キャッシュヒット率を変えて評価
  - 最悪回転数は針が CLOCK リストを一周する場合なので自明
  - キャッシュヒット時は参照ビットを1にするだけなので自明
- CLOCK と同等の時間計算量を達成
  - 数百回の回転を要する CLOCK-Pro と比較して大幅な削減

キャッシュミス時の針の平均回転数

キャッシュヒット率	CLOCK	CLOCK-Pro	CUSH	Compact CAR
0.30	1.14	9.52	1.45	3.52
0.50	1.20	16.11	1.47	3.68
0.70	1.32	**190.44**	1.51	3.78
0.90	1.73	4.20	5.14	3.20

まとめと今後の課題

- まとめ
  - ICN ルータでの実装を考慮したキャッシュ置換方式を提案
    - scan 耐性と loop 耐性によってネットワークへの適正を実現
    - CLOCK-Pro のオーバーヘッドを削減して低コストに拡張可能なデータ構造とアルゴリズムを実現
  - 提案方式のネットワークトラフィックへの適正を評価
    - CUSH は元となった CLOCK-Pro と同等の性能を達成
    - 単純な方式ではキャッシュヒットできないアクセスパターンに対処可能
- 今後の課題
  - CUSH のハードウェアアーキテクチャの考察
  - キャッシュ配置・判断方式と組み合わせた場合の性能評価

CUSH の特徴

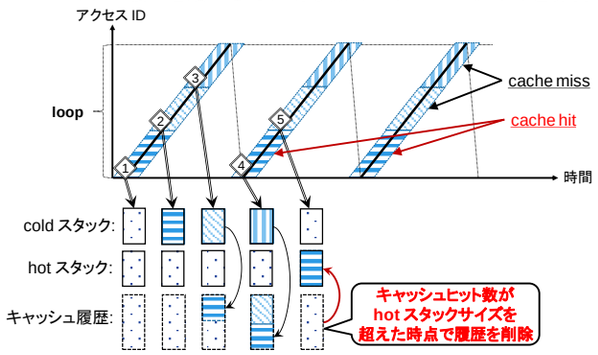
- ネットワークへの適性
  - scan 耐性を実現し、ワンタイムアクセスに強い
    - 履歴用ハッシュテーブルを2つ交互に用いることで hot チャンクの検出性能も向上
  - loop 耐性を実現し、チャンク単位アクセスに強い
    - loop 耐性向上のための履歴拡張コストが小さい
    - キャッシュエントリと履歴エントリを分離しても loop 耐性を維持できるアルゴリズムを提案
  - 人気の頻繁な移り変わりに強い
    - 人気の頻繁な移り変わりに弱い CLOCK-Pro からアルゴリズムを改善
- ルータでの実装オーバーヘッド
  - CLOCK と同等のメモリ・計算オーバーヘッド
  - 履歴実装のためのオーバーヘッドを大幅削減

loop 耐性実現のための適切な履歴削除

- CUSH はキャッシュと履歴を分離したため、履歴の削除を行う機構を引き継げない
  - 適切なタイミングで履歴を削除しなければ hot チャンクが溢れてしまう
  - LIRS では LRU スタックの順序によって管理
  - CLOCK-Pro では CLOCK リスト順序と test flag によって管理
- ヒット数に応じて履歴を削除することで動作を再現
  - hot チャンク数と同数のヒットがあった場合に履歴を削除

CUSH の loop 耐性

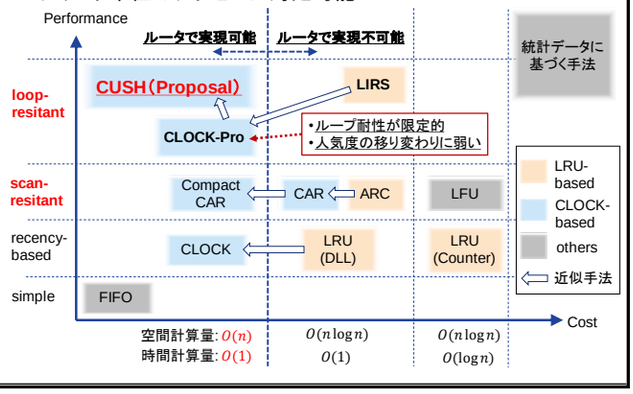
- キャッシュ履歴を活用して適切な数の hot チャンクのみを保持



Osaka University 30

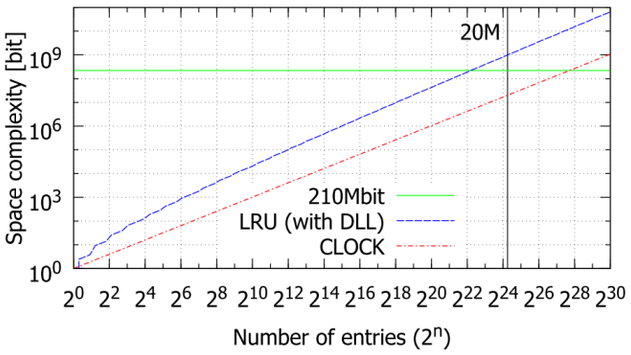
提案方式の位置づけ

- チャンク単位のアクセスに対処可能



Osaka University 31

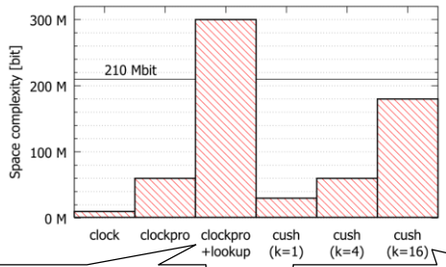
LRU と CLOCK のオーバーヘッドの比較



Osaka University 32

CUSH の空間計算量オーバーヘッド

- 処理速度の要件から SRAM (最大 210 Mbit)を利用
- 既存研究で実現可能とされる10^7エントリを想定



キャッシュ履歴による検索機構へのオーバーヘッドを含めると、CLOCK-Pro は SRAM の容量を超過

CUSH は履歴を 16倍まで拡張しても SRAM の最大容量内に収容可能