博士論文

# 追い出し不可属性を持つ<br>キャッシュアルゴリズムの研究

( Stubborn Cache: Cache with Non-Replacing Lines )

東京大学大学院 情報理工学系研究科
電子情報学専攻

野村 隼人

# 概要

メモリサブシステムによるデータ供給能力はプロセッサの性能上最も重要な要素の一つであり，これを支えるため近年のプロセッサは大容量の Last Level Cache (LLC) を備えている．キャッシュ容量が大きくなることで，プログラマが意識せずとも得られるプログラムのメモリアクセス性能が良くなっている．しかしながら，大容量化した LLC には挿入後追い出されるまで再度利用されることのないようなキャッシュラインであるデッドブロックが多く含まれており，その一方でなおキャッシュミスは多く生じている．これは，直近の履歴やその学習によって得られるキャッシュラインの保持の選択を行うような従来のアプローチが短期の再参照のヒット率を高めることを指向しているためで，特に memory-intensive なアプリケーションでは，メガバイトオーダーの容量をもつ LLC でも対処できないミスが生じている．大容量化した LLC をさらに活用するためには，従来の手法がターゲットとしていたものより再参照までの距離間隔（時間）がより長いキャッシュラインへのアクセスや，より大きいワーキングセットをターゲットとしたキャッシュマネジメントが必要となる．

本研究による調査では，まず，LLC で生じるキャッシュミスの多くを占めるものが再参照ミスであり，また 10〜100M（メガ）の長期の命令数間隔に渡る再参照であることを明らかにした．これは周期の長いスラッシングや，ストリーミングアクセスによってもたらされている．

本研究ではそのようなミスに対処するためのキャッシュマネジメント手法として，Stubborn 戦略を提案した．これは，キャッシュラインの追い出しを一時的に凍結し，キャッシュラインの入れ替えを起こさない領域を設けることで，長期保持による統計的なヒット数向上を可能とするような戦略である．この Stubborn 戦略の最も単純な実装である Normal Stubborn では，従来の手法では的確な追い出し選択が困難であった 10〜100M 命令間隔の長期再参照が連続するワークロードや，激しいスラッシングによりキャッシュ容量が全く活用されないワークロードなどに有効であることを確認した．さらに，ワークロードのメモリアクセス傾向を詳細に調べ，モデルと共に例示することで Stubborn 戦略が有効であったワークロードにおいて，キャッシュラインの凍結という一見非効率な Stubborn 戦略が何故 LLC において有効に機能するかを明らかにした．

Stubborn 戦略は，従来の置き換えアルゴリズムとハイブリッドに実装され，キャッシュの一部の置き換えをアクセス履歴に関係なく凍結する．この性質から，もし不要なキャッシュラインを Stubborn 戦略で取り込んでしまった場合，キャッシュの容量効率を純減させることとなる．Normal Stubborn のようなナイーブな実装ではこの問題による副作用が確認された．このため，プログラムの動作傾向，ワークロードに合わせて，どのキャッシュラインを，いつからいつまで Stubborn 戦略で運用するかを動的に判断

する機構の実現が Stubborn 戦略の実用上の課題となる．

そこで本論文では，Stubborn 戦略が多様なワークロードに適用するための条件について調査し，さらなる性能向上と副作用の抑制を実現する方法を明らかにする．この調査では，Stubborn 戦略のキャッシュ容量への依存性， フェーズに適応させたときの Stubborn 戦略の効果，Stubborn 領域の静的な増減による効果，全域を Stubborn 領域としたときの効果について調べ．フェーズ適応による副作用の抑制と，Stubborn 領域の増減による性能向上が得られることを確認した．また，それに基づいて実行時ワークロード適応型の Stubborn キャッシュマネジメント手法である Stubborn-DYN, BRRIP triggered Stubborn，Stubborn-HL を提案した．

シミュレータ上にこの提案手法 Stubborn-DYN，BRRIP triggered Stubborn，Stubborn-HL, Stubborn-HL-Half, Stubborn-HL-Reset を実装した評価では，LRU に対して提案手法 Stubborn-HL-Reset により最大 42.3%, 幾何平均で 3.8%の性能向上を確認した. また, Normal Stubborn と比較した場合, 今回提案した適応制御により 2.8%の性能向上となり，特に，7.4%性能低下していたワークロードでは性能低下を抑えて 0.1%の性能向上が得られるまで改善が見られるなど，性能を安定させる効果が確認できた．

この結果より，本研究の成果として，長期保持による統計的なヒット数向上を可能とするような戦略である Stubborn 戦略とその適応的な拡張手法によって，大容量化した LLC を従来の手法とは異なるアプローチで活用することでメモリサブシステムによるデータ供給能力を高め，プロセッサの性能向上に寄与することを示した．

# 目次

# 図目次

# 表目次

# 第1章 序論

## 1.1. 研究背景と目的

メモリサブシステムによるデータ供給能力はプロセッサの性能上最も重要な要素の一つであり，これを支えるため近年のプロセッサは大容量の Last Level Cache (LLC) を備えている[1]–[6]．キャッシュ容量が大きくなることで，プログラマが意識せずとも得られるプログラムのメモリアクセス性能が良くなっている．また，キャッシュミスを削減し，性能を向上させるために様々なキャッシュアルゴリズムが研究・提案され続けている．プリフェッチングついてはスロットリング手法の研究によって適切なプリフェッチ量を調整できるようになり，新しい置き換えアルゴリズムも盛んに研究されている．

しかしながら，大容量化した LLC には多くのデッドブロックが含まれており，その一方でキャッシュミスはなお多く生じていることが知られている[7]–[9]．これは，直近の履歴やその学習によって得られるキャッシュラインの保持の選択を行うような従来のアプローチが短期の再参照のヒット率を高めることを指向しているためで，特に memory-intensive なアプリケーションでは，メガバイトオーダーの容量をもつ LLC でも対処できないミスが生じている．

大容量化した LLC をさらに活用するためには，従来の手法がターゲットとしていたものより再参照までの距離間隔（時間）がより長いキャッシュラインへのアクセスや，より大きいワーキングセットをターゲットとしたキャッシュマネジメントが必要となる．

本研究による調査では，まず，LLC で生じるキャッシュミスの多くを占めるものが再参照ミスであり，また 10～100M（メガ）の長期の命令数間隔に渡る再参照であることを明らかにした．これは周期の長いスラッシングや，ストリーミングアクセスによってもたらされている．

そこで，本研究ではそのようなミスに対処するためのキャッシュマネジメント手法として，Stubborn 戦略を提案した．これは，キャッシュラインの追い出しを一時的に凍結し，キャッシュラインの入れ替えを起こさない領域を設けることで，長期保持による統計的なヒット数向上を可能とするような戦略である．この Stubborn 戦略の最も単純な実装である Normal Stubborn では，従来の手法では的確な追い出し選択が困難であった 10～100M 命令間隔の長期再参照が連続するワークロードや，激しいスラッシングによりキャッシュ容量が全く活用されないワークロードなどに有効であることを確認した．さらに，ワークロードのメモリアクセス傾向を詳細に調べ，モデルと共に例示する

ことで，Stubborn 戦略が有効であったワークロードにおいてキャッシュラインの凍結という一見非効率な Stubborn 戦略が何故 LLC において有効に機能するかを明らかにする．

Stubborn 戦略は，従来の置き換えアルゴリズムとハイブリッドに実装され，キャッシュの一部の置き換えをアクセス履歴に関係なく凍結する．この性質から，もし不要なキャッシュラインを Stubborn 戦略でセットに留めてしまった場合，キャッシュの容量効率を純減させることとなる．Normal Stubborn のようなナイーブな実装ではこの問題による副作用がある．このため，プログラムの動作傾向，ワークロードに合わせて，どのキャッシュラインを，いつからいつまで Stubborn 戦略で運用するかを動的に判断する機構の実現が Stubborn 戦略の実用上の課題となる．

そこで本論文では，Stubborn 戦略が多様なワークロードに適用するための条件について調査し，さらなる性能向上と副作用の抑制を実現する方法を明らかにする．この調査では，Stubborn 戦略のキャッシュ容量への依存性，フェーズに適応させたときの Stubborn 戦略の効果， Stubborn 領域の静的な増減による効果，全域を Stubborn 領域としたときの効果について調べ．フェーズ適応による副作用の抑制と，Stubborn 領域の増減による性能向上が得られることを確認した．また，それに基づいて実行時ワークロード適応型の Stubborn キャッシュマネジメント手法を提案した．

シミュレータ上にこの提案手法 Stubborn-DYN，BRRIP triggered Stubborn，Stubborn-HL，Stubborn-HL-Half，Stubborn-HL-Reset を実装した評価では，提案手法 Stubborn-HL-Reset により LRU に対して最大 42.3%，幾何平均で 3.8%の性能向上を確認した．また，Normal Stubborn と比較した場合，今回提案した適応制御により 2.8%の性能向上となり，特に，7.4%性能低下していたワークロードでは性能低下を抑えて 0.1%の性能向上が得られるまで改善が見られるなど，性能を安定させる効果が確認できた．

この結果より，本研究の成果として，長期保持による統計的なヒット数向上を可能とするような戦略である Stubborn 戦略とその適応的な拡張手法によって，大容量化した LLC を従来の手法とは異なるアプローチで活用することでメモリサブシステムによるデータ供給能力を高め，プロセッサの性能向上に寄与することを示した．

## 1.2. コントリビューション

本論文の貢献は以下の 6 点である.

- 従来手法では対処できない，残されたキャッシュミスが長期の再参照により生じるミスであることを明らかにした
- そのような従来手法では対処できないキャッシュミスについて，より長い間隔での再参照の保護を目的とした Stubborn 戦略を提案した
- Stubborn 戦略の最も単純な実装においても，目的とした長期再参照によるミスを減らし，同時に性能向上が得られることを確認した
- 調査により Stubborn 領域と性能の関係について傾向解析を行い，この結果から Stubborn 戦略が有効に機能する条件を明らかにした
- 適応型 Stubborn キャッシュマネジメント手法を提案し，ハードウェアで実現する手法を明らかにした
- 適応型 Stubborn キャッシュマネジメント手法の評価において，性能向上と副作用の抑制の両立を確認した

## 1.3. 用語の定義

本研究では，キャッシュマネジメントの研究で用いられるハードウェア機構とキャッシュメモリアクセスに関する用語を多用する．ここで，予めそれらの用語について解説，定義する．

### 1.3.1. キャッシュブロック配置方式

まず，キャッシュブロックの配置方式に関する用語について解説する．
次の図 1 はセット・アソシアティブな構成のキャッシュテーブルにおける一例を，ハードウェア機構を簡略化して図式化したものである．

**ライン**
キャッシュメモリにおけるデータの取り扱いの最小単位であり，プロセッサの設計指向や年代によっても異なるが概ね 64 byte 程度のサイズの領域を持つ．キャッシュメモリへのデータの挿入，追い出しはこのラインが単位となる．
**セット**
データのアドレスから抽出された index に沿ってラインを挿入・参照する際のインデックスとなる数字で，キャッシュテーブルの行の単位である．
**ウェイ**
1 セットあたりの列を意味し，同時に 1 セット中の特定のラインを指すためのインデックスとしても使用される．1 セットの中に格納されるライン数により，この図では 4 way キャッシュとなる．

本論文では，セット中の特定のラインの挿入位置を意味する場合 way n（n は 8 way キャッシュテーブルであれば 0～7），1 セットあたりのライン数またはある手法の適用範囲が 1 セットに占めるウェイの数を m way（m は 8 way キャッシュテーブルであれば 0～8）と表記する．

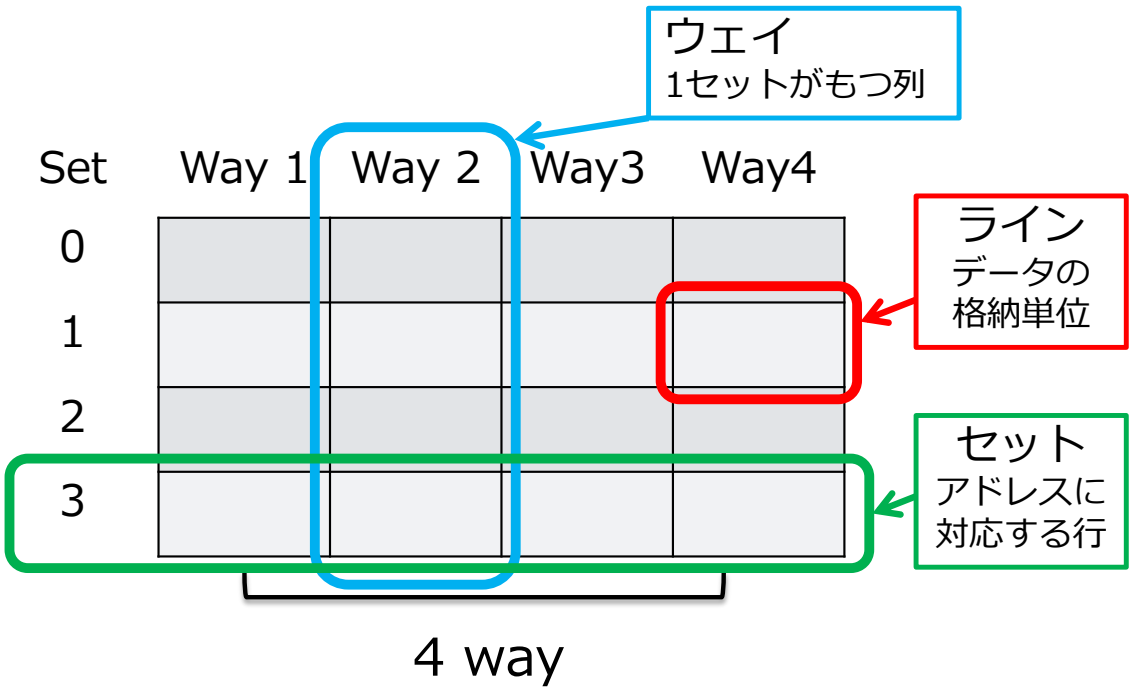



図 1 セット・アソシアティブな構成のキャッシュテーブル

また，セット・アソシアティブな構成のキャッシュテーブルにおけるキャッシュアクセスの様子を次の図 2 に図解する．この図では，キャッシュへの Read 要求が発生したとき，その要求データのアドレスについて index を元にキャッシュテーブルのセットを定め，そのセット内に目的のアドレスのデータが存在するかどうかを tag の照合により判断している様子を示している．このように，index で指定されたセット内に tag が一致するラインがあれば，そのラインが要求していたデータを持つラインであることがわかり，キャッシュヒットとなる．そして，その合致したラインの持つデータを CPU コアや上位キャッシュに提供する．ここでは valid ビットなどによる判断や追い出しオーダーのフラグとその更新の処理については省略している．

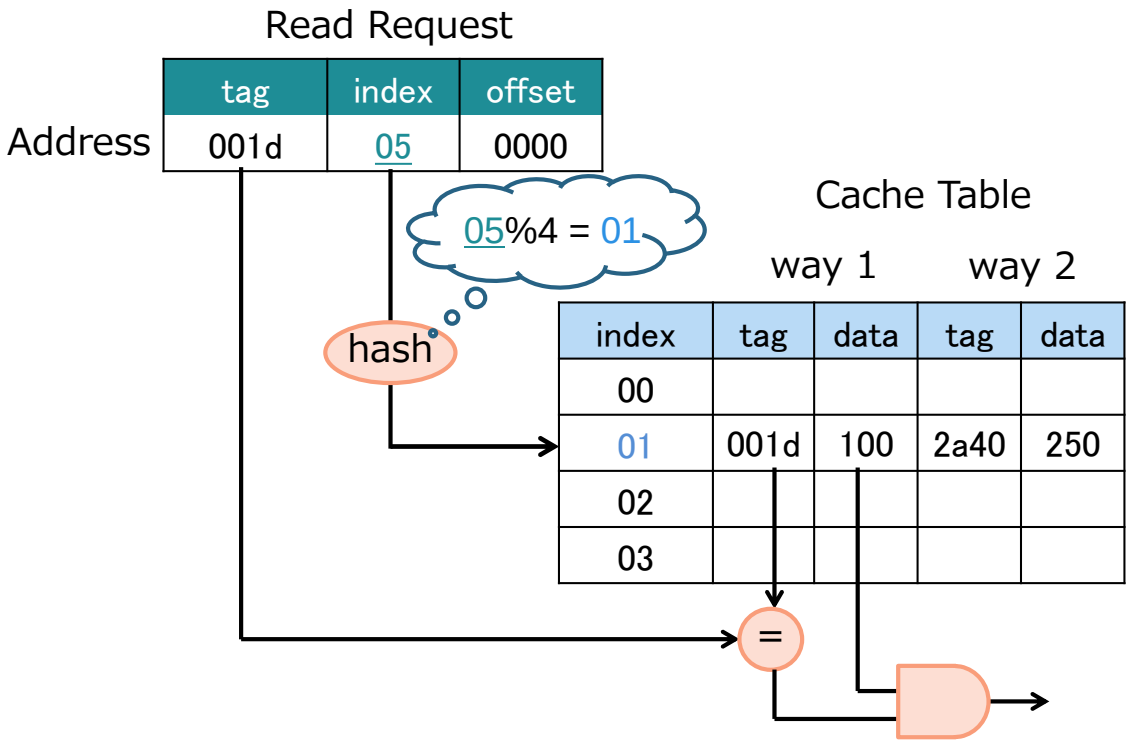



図 2 セット･アソシアティブな構成のキャッシュテーブルへのアクセス

### 1.3.2. キャッシュアクセス

#### 1.3.2.1. 初期参照と再参照

プログラム実行中に生じるアクセスについて，プログラム実行期間中あるいはある一定の期間における，あるアドレスへの最初のアクセスを初期参照と呼び，そのアドレスへの 2 度目以降のアクセスを再参照と呼ぶ．あるアドレスへのアクセスから次の同じアドレスへのアクセスの間に追い出しと再挿入が生じたラインについても，これを再参照とする．プリフェッチによって初めてアクセスされるアドレスについては，プリフェッチによる挿入時とその要求が初期参照となる．

#### 1.3.2.2. 再参照距離

「再参照距離」「再参照間隔」といった語について，研究により定義が異なる場合があるが，本研究では「あるアドレスによって挿入・参照されるキャッシュラインへのデマンドアクセスについて，同一のキャッシュラインへの前回のデマンドアクセスからの距離（実行命令数）」として再参照距離を定義する．再参照距離の例を次の図 3 に示す．

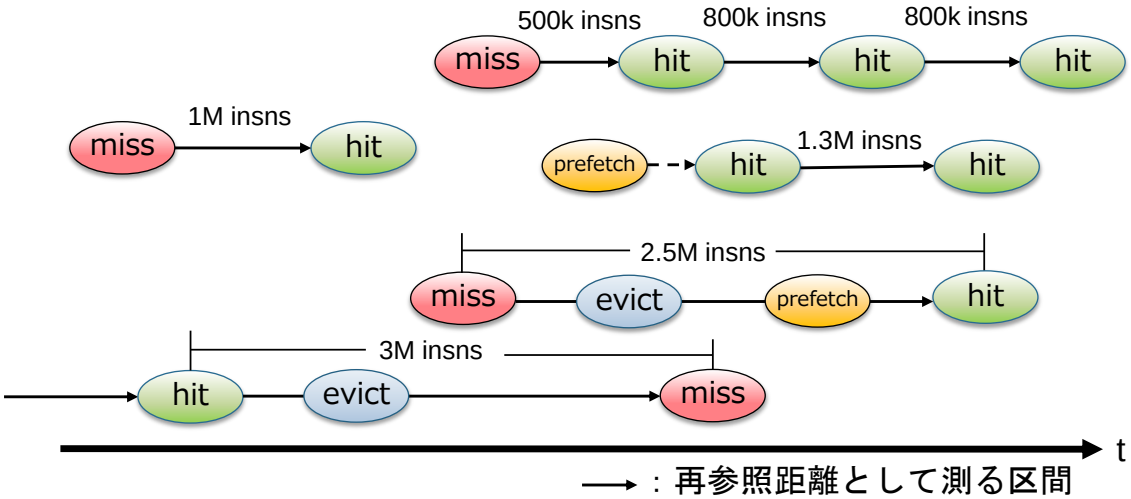



図 3 再参照距離の例示

### 1.3.2.3. スラッシング

スラッシングは，キャッシュサイズ（単一セット内ではウェイ数）を上回るデータ量により玉突きの要領で追い出しの連鎖を発生させるようなキャッシュアクセスのパターンである．そのアクセスではそれぞれのデータは再参照があるにも関わらず，次回参照時までにはキャッシュから追い出されている（図 4）．

後述のベンチマークの中では xalancbmk（XML パーサ），omnetpp（ネットワークシミュレータ）の多くのアクセスパターンがこれに相当する．例として xalancbmk での実際のアクセスパターンを図 5 に示す．このグラフは，縦軸がアドレス空間，横軸が実行命令数を指し，実行命令数が大きくなっているほど時間が経過しているとして読む．それぞれの点のプロット 1 つ 1 つがキャッシュメモリへのアクセスであり，緑の点はそのアクセスがヒット，赤はミスしたことを意味する．

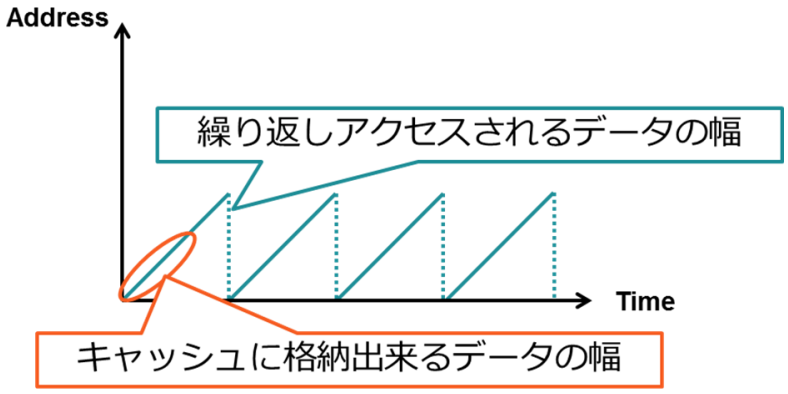



図 4 スラッシング

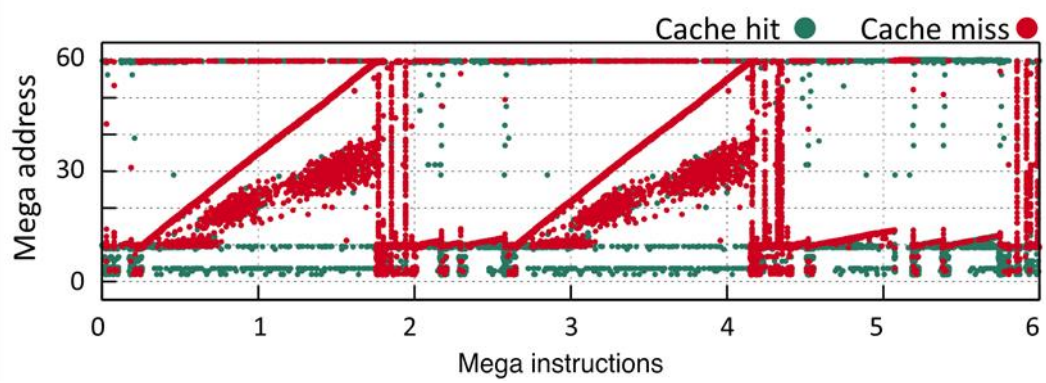



図 5 スラッシングの実例(483.xalancbmk)

### 1.3.2.4. ストリーミング

ストリーミングは，一度しかアクセスされないデータへの連続的なアクセスか，再参照があっても従来の置き換えアルゴリズムではとても対応できない程度に再参照間隔が極端に長いような連続したアクセスを指す（図 6）.

後述のベンチマークの中では GemsFDTD（計算電磁気学計算），libquantum（量子力学計算），mcf（組み合わせ最適化）の多くのアクセスパターンがこれに相当する[10]. 例として mcf での実際のアクセスパターンを図 7 に示す.

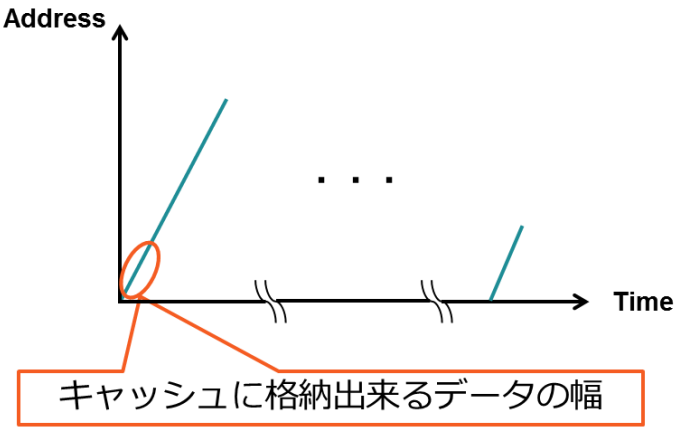



図 6 ストリーミング

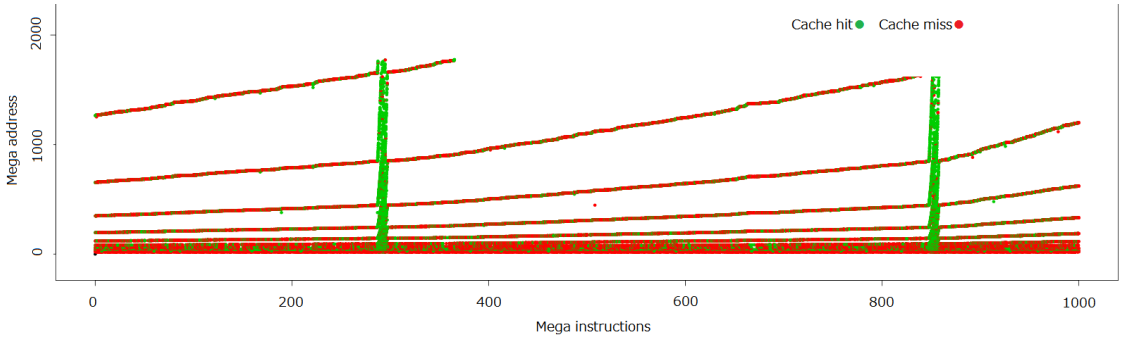



図 7 ストリーミングの実例(429.mcf)

## 1.4. 評価環境

本論文では，以降置き換え戦略の解析を行う第 3 章から第 7 章のまとめに渡って，様々な予備調査と評価を行う．評価では共通した評価環境を用いる．その評価環境を予めここに記す.

### 1.4.1. ベースラインプロセッサ

Alpha ISA[11]–[13]のサイクルアキュレートなプロセッサシミュレータプロセッサシミュレータ鬼斬弐[14], [15]にベースラインとなる置き換えアルゴリズム，提案置き換えアルゴリズム，プリフェッチャを実装し，サイクルレベルの評価を行う．プロセッサの構成は表 1 に示した値を用いた. この構成は Jaleel らによる評価環境[16]に合わせたものである．シングルコアシングルスレッド実行で，プリフェッチの適用は LLC のみとした．プリフェッチャには Srinath らによるストリームプリフェッチャ[17]を用いる．

特に断りがなければ，以降の性能評価においてベースとするシミュレータのアーキテクチャパラメタは以下の通りとする. LLC で使用する置き換えアルゴリズムについては随時指定する.

表 1 標準ベースラインプロセッサパラメータ

| | |
|---|---|
| Processor | |
| Core | Alpha ISA, single core, single thread |
| Issue width | int:2, fp:2, mem:2 |
| Inst. window | int:32, fp:16, mem:16 |
| Branch pred | 8KB g-share |
| BTB | 2K entry, 4way |
| LSQ | 96 entry |
| Cache Memory | |
| I/D L1 Cache | LRU, 32 KB, 4 way, 64 B line, 3 cycle latency |
| L2 Cache | LRU, 512 KB, 8 way, 64 B line, 10 cycle latency |
| L3 Cache (LLC) | 2MB, 8 way, 64 B line, 24 cycle latency<br>Stream prefetcher (degr:16, dist:16) |
| Memory access | 250 cycle latency |

### 1.4.2. 実行ベンチマーク

ワークロードについてはSPEC CPU 2006 Benchmark Suite[18] 29本中から，以下のmemory-intensiveなワークロード12本を評価に使用する．

- bwaves：流体力学
- mcf：組み合わせ最適化（ネットワークシンプレックス法アルゴリズム）
- milc：物理学／量子色力学
- leslie3d：流体力学
- soplex：線形計画法，最適化
- GemsFDTD：計算電磁気学（時間領域差分法）
- libquantum：物理学／量子計算（ショアの多項式時間因数分解アルゴリズム）
- lbm：流体力学（格子ボルツマン法）
- omnetpp：離散イベントシミュレーション
- astar：経路探索アルゴリズム(A*)
- sphinx3：音声認識
- xalancbmk：XML 処理

各ベンチマークの概要については富士通発行のホワイトペーパー，『ベンチマークの概要 SPECcpu2006』[19]より引用．このラインナップは，上記表 1 のアーキテクチャパラメタにおいてLLCの置き換えアルゴリズムをLRUとし，プリフェッチを適用していない構成で予備評価を行い，LLCのMPKI (Miss per Kilo-Instruction)が1以上になるものを抽出したものである．シミュレーションでは先頭 10G 命令スキップ後，続く1G命令の区間についてcycle accurateに実行し，測定する．

## 1.5. 論文の構成

本論文は全 7 章で構成される．本論文の各章の構成を図 8 に示す.

2 章では関連研究について紹介し，それぞれの関係について述べる．3 章では従来手法に残された課題を整理し，その結果を元に Stubborn 戦略を提案，評価する．4 章では Stubborn 戦略の効果を解析し，さらなる性能向上のための道筋を立てる．4 章での調査結果を元に 5 章で Stubborn 戦略の適用タイミングを動的に判断する手法を提案，評価し，6 章で Stubborn 戦略の領域の動的に増減させる手法について提案，評価する．7 章では本研究の成果をまとめ，今後の展望を述べる.

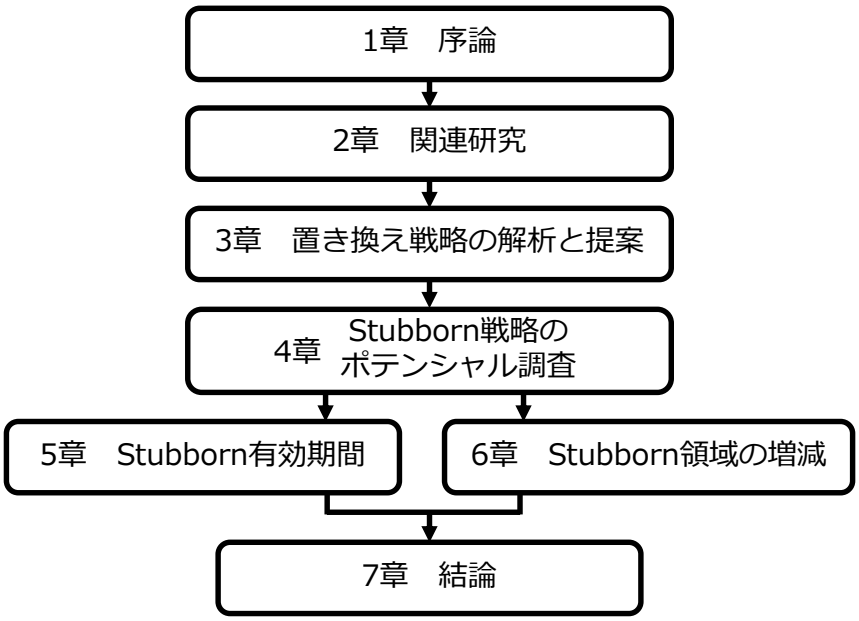



図 8 論文の構成

# 第2章 関連研究

この章では，既存のキャッシュアルゴリズムについて述べる．キャッシュアルゴリズムのなかで特に重要な 2 つの技術の枠組みとして，置き換えアルゴリズムとプリフェッチが存在する．関連研究として，本研究においてベースラインとして利用する置き換えアルゴリズムから，最新の置き換えアルゴリズムおよびプリフェッチの一部までを紹介し，それぞれの立ち位置と関係を述べる．

## 2.1. 置き換えアルゴリズム

置き換えアルゴリズムは，キャッシュに残されるべきデータを，過去のアクセス履歴を用いて各キャッシュラインの重要度（再参照可能性と再参照までの近さ）を推定し，判断することで時間局所性のあるプログラムのキャッシュアクセスを支援する．

### 2.1.1. OPT

Belady's OPT とも呼ばれる OPT[20]は，置き換えアルゴリズムの理論的な最適動作を示すアルゴリズムで，これは未来のアクセスを全て見通せることを前提に次回のアクセスまでの時間が最も長いラインを追い出す（図 9）．この条件のため，実プロセッサでは完全には再現できない．

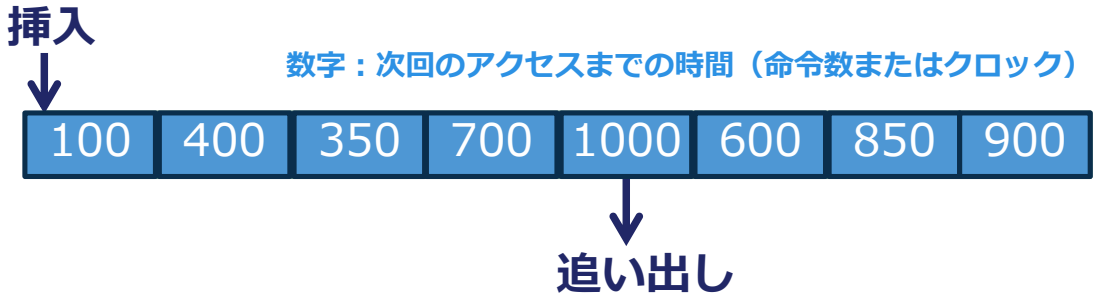



図 9 OPT における追い出し操作

### 2.1.2. LRU

LRU (Least Recently Used)[21]は代表的なキャッシュ置き換えアルゴリズムであり，様々な研究報告がこれをベースとして性能向上を報告している．LRU では，最も最近参照されたキャッシュラインおよび挿入直後のキャッシュラインを MRU (Most Recently Used)，新たな挿入および再参照により最も参照のないキャッシュラインとなったものを LRU とし，新たな挿入が発生したときに LRU にあるキャッシュラインが追い出すことで，時間局所性のあるプログラムのキャッシュアクセスを支援する（図 10）．しかしながら，過去に再参照のあったキャッシュラインを新たに挿入されたキャッシュラインと区別せずに Most Recently Used の状態とするため再参照のあるキャッシュラインの保護が弱く，直近の履歴に惑わされてしまう．具体的には再参照頻度の高いアクセスがプリフェッチやスラッシング，ストリーミングアクセスにより追い出されてしまう．

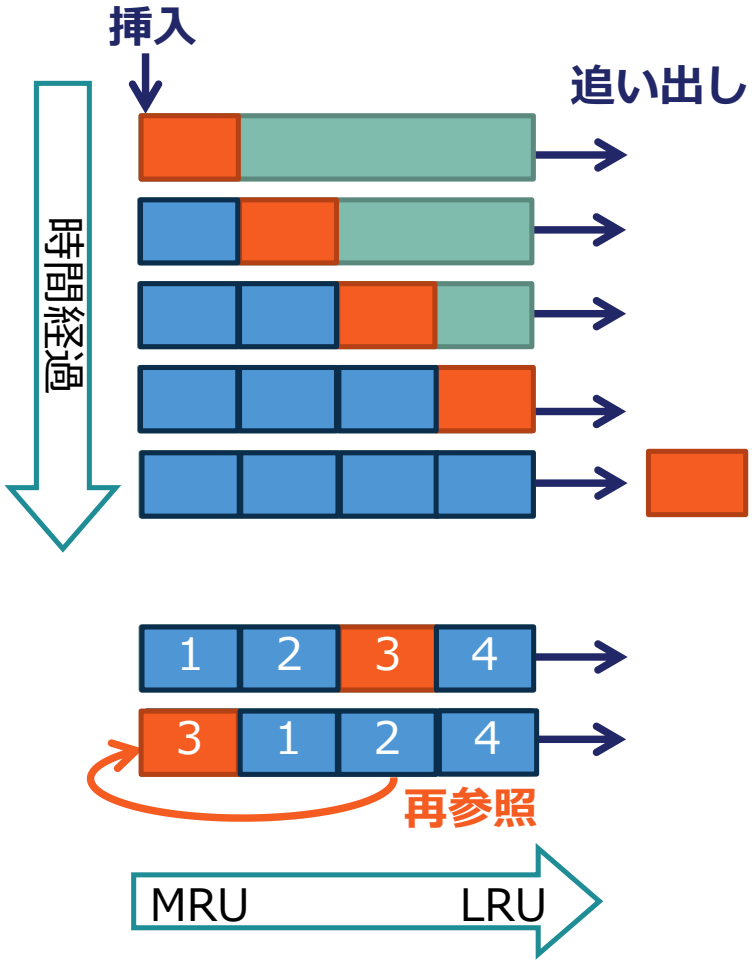



図 10 LRU における挿入・追い出しの操作

### 2.1.3. RRIP

Jaleel らが提案する Re-Reference Interval Prediction (RRIP)[16]では，追い出し優先度の設定において，再参照のあったキャッシュラインの保護を優先する．RRIP では LRU オーダーから替ってその再参照頻度を学習するためのカウンタ，RRPV (Re-Reference Prediction Values)を持つ．RRPV は挿入時に RRPV が取れる値の中間値（2bit の RRPV であれば 0～3 のうち 2）で挿入され，再参照があれば 0 でリセットされる．RRPV は時間経過でインクリメントされ，RRPV が最大値のキャッシュラインが追い出されることで，セット中に保持している間に再参照のあったキャッシュラインを保護している．この手法を SRRIP という．SRRIP での挿入・追い出しの操作を図 11 に示す．

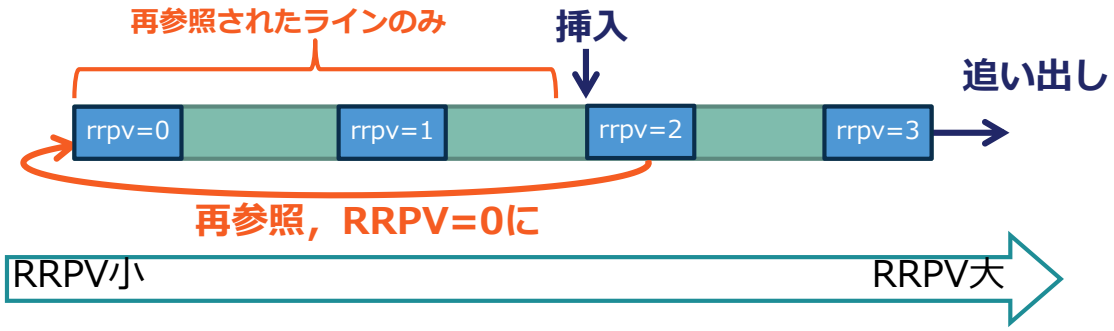



図 11 SRRIP における挿入・追い出しの操作

また，この研究ではスラッシングに耐性を持たせるため，RRPV の初期値をランダムに変更する手法 BRRIP を提案している．BRRIP での挿入・追い出しの操作を図 12 に示す．このように，ランダムに相対的に優遇するラインを発生させることで，スラッシング発生時に全てのキャッシュラインが追い出されてしまうことが防げる．

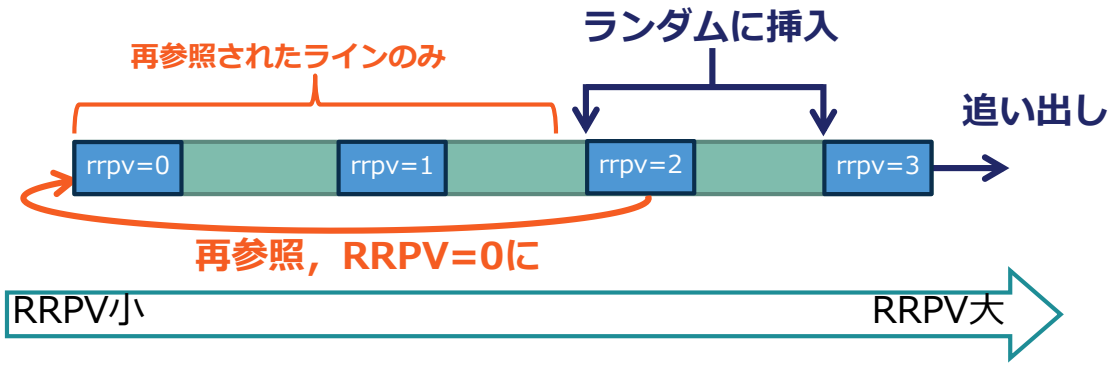



図 12 BRRIP における挿入・追い出しの操作

また Jaleel らは SRRIP と BRRIP を動的に切り替えて使用する DRRIP を同時に提案している．SRRIP と BRRIP はそれぞれワークロードの実行フェーズごとに優劣がつくため，どちらを適用するかを動的に判断するための手法として，RRIP の前身的手法である DIP[22], [23]から Set Dueling Monitor (SDM)が用いられている．SDM は複数の手法についてモニタリングセットでそれぞれ実行し，その成績を PSEL (Policy Selector)カウンタで集計してどちらの手法が優秀かを判断し，フォロワーセットに適用する（図 13）.

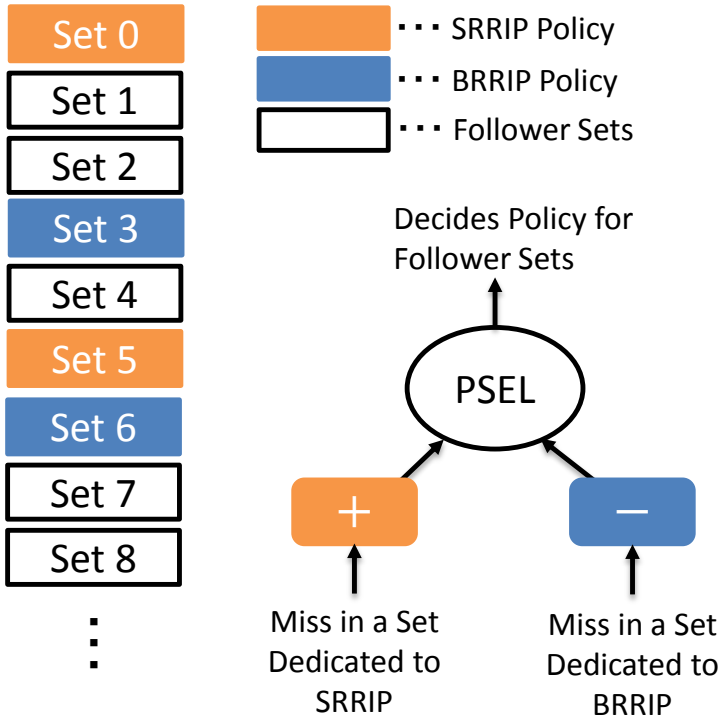



図 13 DRRIP における SDM

RRPV を用いた追い出し優先度管理の LRU オーダーとの違いは，この RRPV は複数のラインで同じ値をとることがある点である．場合によってはすべてのラインで同じ値をとることがあり，そのような場合再参照により一度 RRPV が 0 となったラインもすぐに RRPV が最大値となり追い出される可能性が高くなる．

このように，RRIP は再参照アクセスの保護を意識した置き換えアルゴリズムであるが，再参照のあったキャッシュラインの学習を保持できる期間はそのキャッシュラインが挿入されてから追い出されるまでの間に限られ，BRRIP で対処できない量のスラッシングが発生した際は再参照のあったキャッシュラインの RRPV 値もインクリメントされていくことで結果的に追い出されてしまう．

### 2.1.4. PACMan

Wu らによる Prefetch-Aware Cache Management (PACMan)[24]は，挿入時にデマンドによるものかプリフェッチによるものかを区別し，プリフェッチにペナルティを与える, 具体的には挿入時の RRPV を最大値とすることでデマンドでしか挿入されないキャッシュラインの保護の強化を目的とした置き換えアルゴリズムである．

### 2.1.5. SHiP

DRRIP では SRRIP，BRRIP の切り替えによる RRPV の初期値変更を行い，これによりキャッシュラインごとの再参照予測としていた．これに対し，Wu らによる Signature-Based Hit Predictor (SHiP)[25]では， SHiP と呼ばれる PC ベースの学習による再参照予測器により，挿入時に再参照の可能性を予測し，RRPV の初期値を判断している．発展型として，学習時のプリフェッチ対応を施した SHiP++[26]がある．

### 2.1.6. Hawkeye

Jain らによる Hawkeye[27]は再参照予測において OPT を模した予測器を用いた手法である．真の OPT の再現には全ての未来のアクセスを事前に知っている必要があるため実プロセッサでは再現できないが，Hawkeye における学習機 OPTgen では過去の時点から見て現在までのアクセス履歴を用いることで OPT の動作を模し，その判断を用いて現在から未来でも同様の結果が得られるものとして利用している．OPT の振る舞いについてプリフェッチを考慮した PA-Hawkeye (Prefetch aware Hawkeye)[28]がある．

## 2.2. プリフェッチ

プリフェッチ（本論文ではソフトウェアでの指示に依らない，ソフトウェアの指示を必要としない，CPU が持つハードウェアプリフェッチャによる動作と手法を指す）も，キャッシュマネジメントにおいて主要なアプローチの一つである．これは，投機的なキャッシュメモリ活用であり，デマンド要求の前に予めアクセスを予測してキャッシュにデータを読むことでメモリレイテンシを隠蔽する手法である．これらの手法は共通して直近の範囲でアクセスするアドレスの空間的局所性に着目して動作する戦略を採る．

### 2.2.1. ストリームプリフェッチ

ストリームプリフェッチ[17], [29]は，ストリームアクセスを予知可能なアクセスと見なし，その連続性を利用してプログラムがそのアドレスを必要として要求する前に予めキャッシュに載せておく手法である．これにより，ストリームアクセスによって生じる初期参照ミスを隠蔽し，性能向上を図る．ストリームプリフェッチの概要図を図 14 に示す．

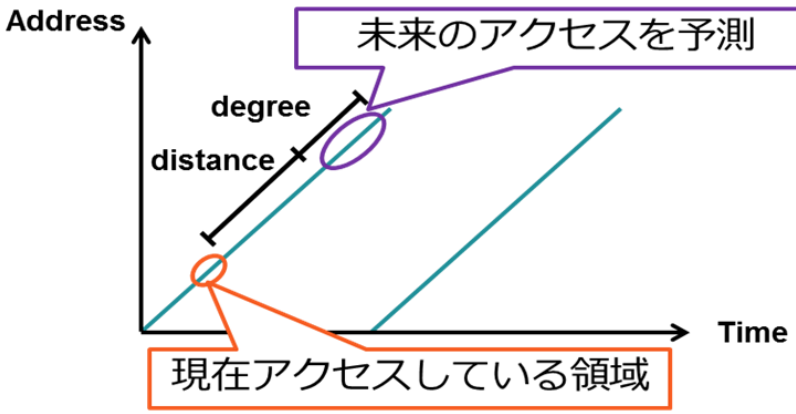



図 14 ストリーミングアクセスとストリームプリフェッチ

### 2.2.2. ストライドプリフェッチ

また，ストリームプリフェッチの変形としてストライドプリフェッチ[30]が存在する．ストライドアクセスというのは，規則的かつ断片的なアクセスのことである．このアクセスを予測して，前もってキャッシュに載せるのがストライドプリフェッチである．

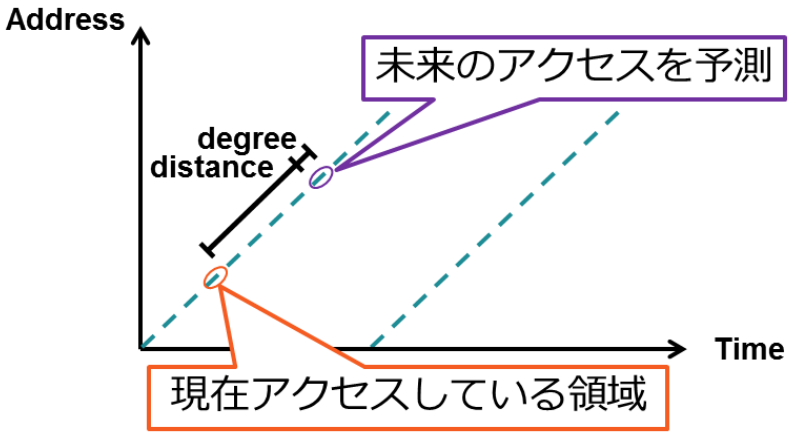



図 15 ストライドアクセスとストライドプリフェッチ

## 2.2.3. GHB プリフェッチ

ここまで紹介したストリームプリフェッチ，ストライドプリフェッチは，単純な線形のアクセスにしか対応できないものであった．それに対して，Nesbit らは複雑なアクセスパターンに対応できるプリフェッチ手法，GHP プリフェッチ[31]を提案している．この実現に Global History Buffer (GHB) と呼ばれるアクセス履歴記憶機構を用いる．GHB の実体は FIFO 構造のテーブルであり，キャッシュミスしたアドレスと，そのアドレスに紐ついて，そのアドレスでミスが生じたときに，次にどのアドレスでミスするかを示すポインタを記録している．これにより，あるアドレスでのミスをトリガーとして，その後再度起こるであろうミスを生じるアドレスを，履歴を元に予測できるようになる．このポインタの連なりを元にプリフェッチを行う．

## 2.2.4. AMPM プリフェッチ

ストリーミングプリフェッチやストライドプリフェッチのような従来のプリフェッチではアクセス予測をメモリアクセスや命令アドレスの連続性を用いているが，こうしたプリフェッチャではアウトオブオーダー実行やループアンローディングのための最適化がなされたコードに対応できないことがある．そのようなワークロードの実行時でのプリフェッチに対処する手法に Access Map Pattern Matching Prefetch (AMPM)[32], [33]がある．Map 構造のアクセス履歴記録を用いてパターンマッチングを行い，アクセスの連続性に依存しない Order-Free なプリフェッチを実現している．

### 2.2.5. CCCPO

プリフェッチは予測可能なアクセスパターンをもつワークロード，またその中の一部のフェーズにおいて効率よくメモリレイテンシを隠蔽し，プロセッサの性能向上をもたらす．一方で，予測が困難なアクセスパターンの実行中には，プリフェッチによってキャッシュに乗せられたデータはそのまま使用されることなくデッドブロックとなってしまい，キャッシュの役割を阻害することがある．そのような問題に対処するため，プリフェッチスロットリング[34]という手法が提案されている．これはプリフェッチャが置き換えアルゴリズムに歩み寄ってプリフェッチ量を調整する手法である．

プリフェッチスロットリングの手法のひとつに，Cache-Convection-Control based Prefetch Optimization (CCCPO)[35], [36]がある．この研究において，プリフェッチ量の調整に Cache Convection (CC)値と呼ばれる指標を用いる．この CC 値は，ある一定の期間でのキャッシュ全体でのヒット数を，再参照されたキャッシュラインの数で割ることで求められる．この CC 値が高いときは，再参照が少ないにもかかわらずヒット数が多い，つまり新たに入ってくるデータが多数を占め，かつプリフェッチがよく効いているような状態である．一方で，CC 値が低いときは，データの局所性が高く再参照が多い，あるいは局所性が薄かったとしても，プリフェッチによってヒット数が稼げていない状態である．これにより，CC 値が高いときにはプリフェッチをより積極的に，低いときにプリフェッチ量を抑えることで，適切なプリフェッチ量の制御を行っている．

## 2.3. まとめ

キャッシュミスを削減し，性能を向上させるために様々なキャッシュアルゴリズムが研究・提案され続けている．本章では，ベースラインとして利用する置き換えアルゴリズム，プリフェッチから，最新の置き換えアルゴリズムおよびプリフェッチの一部までを紹介した．置き換えアルゴリズムの関係を次の図 16 にまとめる．

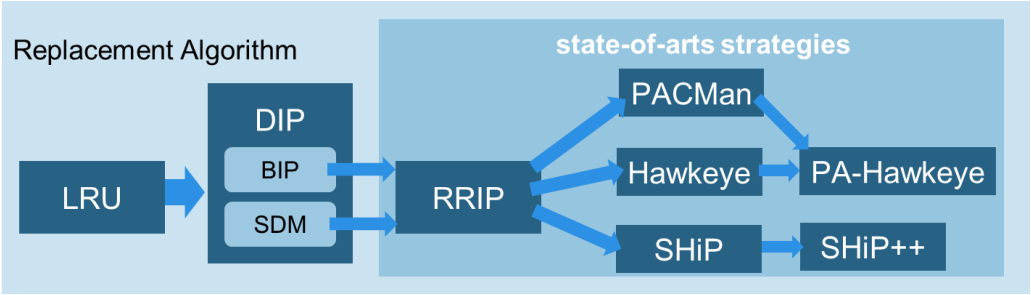



図 16 置き換えアルゴリズムの関係

# 第3章 置き換え戦略の解析と提案

本章では，まず従来手法に残された課題を整理し，調査により置き換えアルゴリズムがとるべき戦略を解析する．そして，その結果を元に従来手法の抱える課題に対応できる置き換えアルゴリズムの戦略，Stubborn 戦略を提案し，最もシンプルな構成での実装として Normal Stubborn を評価する．

## 3.1. 残ったミスは何か

第 2 章では, ベースラインとなる代表的な置き換えアルゴリズム, プリフェッチから，再参照予測を用いる最新の置き換えアルゴリズム，プリフェッチの一部まで紹介した．このように，キャッシュミスを減らす，性能向上を図るため様々なキャッシュアルゴリズムが提案され続けている．

しかしながら，大容量化した LLC には多くのデッドブロックが含まれており，その一方で追い出しによるミスが発生していることが知られている．特に memory-intensive なアプリケーションでは，メガバイトオーダーの容量をもつ LLC でも対処できないミスが生じている．LLC において，それらのキャッシュミスを減らすことによる性能向上の余地がまだある．このポテンシャルについて調査するため，LLC に残ったミスの性質について調査した．

従来のキャッシュアルゴリズムでは比較的直近の履歴から空間的および時間的局所性を抽出している．そこで，本研究では，LLC で生じるミスがこのような局所性に基づくアプローチの改良で解決するか, 再参照間隔に着目した解析を行った．この解析では，残るミスの大勢が初期参照ミスであるのか再参照ミスであるのか，また従来手法で効果がある再参照距離がどの程度であるかを明らかにする．

### 3.1.1. 初期参照と再参照ミス

最初に，依然として生じるミスが初期参照ミスであるのか再参照ミスであるのか調査した．図 17 は，残るミスに占める初期参照ミスと再参照ミスの内訳を示す．縦軸は MPKI で，初期参照ミスと再参照ミスを区別して積み上げている．各ベンチマークで左の棒グラフが LRU，右が DRRIP での MPKI を示す．

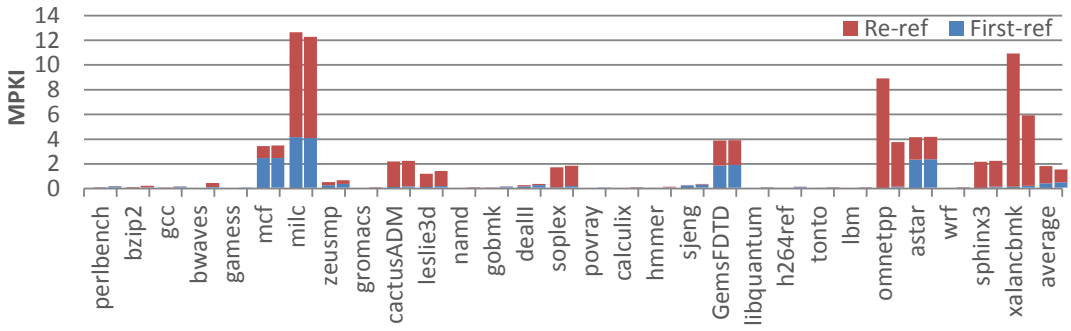



図 17 LLC に残るキャッシュミスに占める初期参照ミスと再参照ミス

結果，SPEC CPU 2006 Benchmark Suite 中の memory-intensive なワークロード 12 本の実行にあたって，プリフェッチを適用した LRU で制御される LLC で生じるミスの 76.3%が再参照ミスであった．DRRIP では再参照時のミス自体が減ったことで 71.4%が再参照ミスである．このことから，対処すべき対象は再参照ミスであると見定めた．

### 3.1.2. 再参照距離

さらに，この再参照ミスはどのような特性をもつのか調査した．調査にあたって，本研究では再参照距離に注目した．図 18 に，再参照ミスに占める，再参照距離ごとのミス数を MPKI で示す．縦軸は，再参照によるミス数を距離毎に分割し積み上げたものを MPKI としたもので，凡例の 100M は 100～999.9…M 命令，10M は 10～99.9…M 命令の再参照距離で生じたミスの集計範囲を意味している．結果，1M～100M オーダーの長距離の再参照によるミスが LLC に残るミスのボリュームゾーンであることがわかった．言い換えると，この調査結果は，「長距離の再参照によって生じるミス」は「発生が稀だから対処しなくて良いミス」ではないことを明らかにした．

また，このグラフから，置き換えアルゴリズムが LRU から DRRIP になったことで対処可能になった長期再参照ミスが見て取れる，それは 1M オーダーでの再参照によるミスである．DRRIP といった再参照予測を用いるキャッシュアルゴリズムやプリフェッチが入った後で生じるのは，このような 10M～100M オーダーの長期間をおいての再参照によるミスである．本論文では，このような長期間をおいての再参照を長期再参照，それによって生じるミスを長期再参照ミスと定義する．この長期再参照ミスの対処は，これまで取り組まれてこなかった．このようなミスの解決が今後キャッシュアルゴリズムによる性能向上のために取り組むべき問題である．

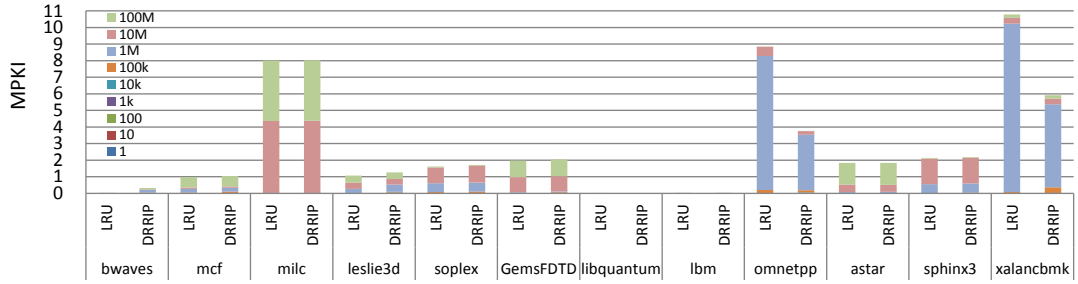



図 18 再参照距離別の再参照ミス数(MPKI)

### 3.1.3. まとめ

本章での調査で, LLC で生じる残されたキャッシュミスの多くを占めるものが再参照ミスであり, また 10〜100M の長期の命令数間隔に渡る再参照であることを明らかにした．これをまとめたものを図 19 に図示する．この結果を踏まえ，大容量化した LLC をさらに活用するためには，従来の手法がターゲットとしていたものより再参照までの距離間隔（時間）がより長いキャッシュラインへのアクセスや，より大きいワーキングセットをターゲットとしたキャッシュマネジメントが必要となると予想する．

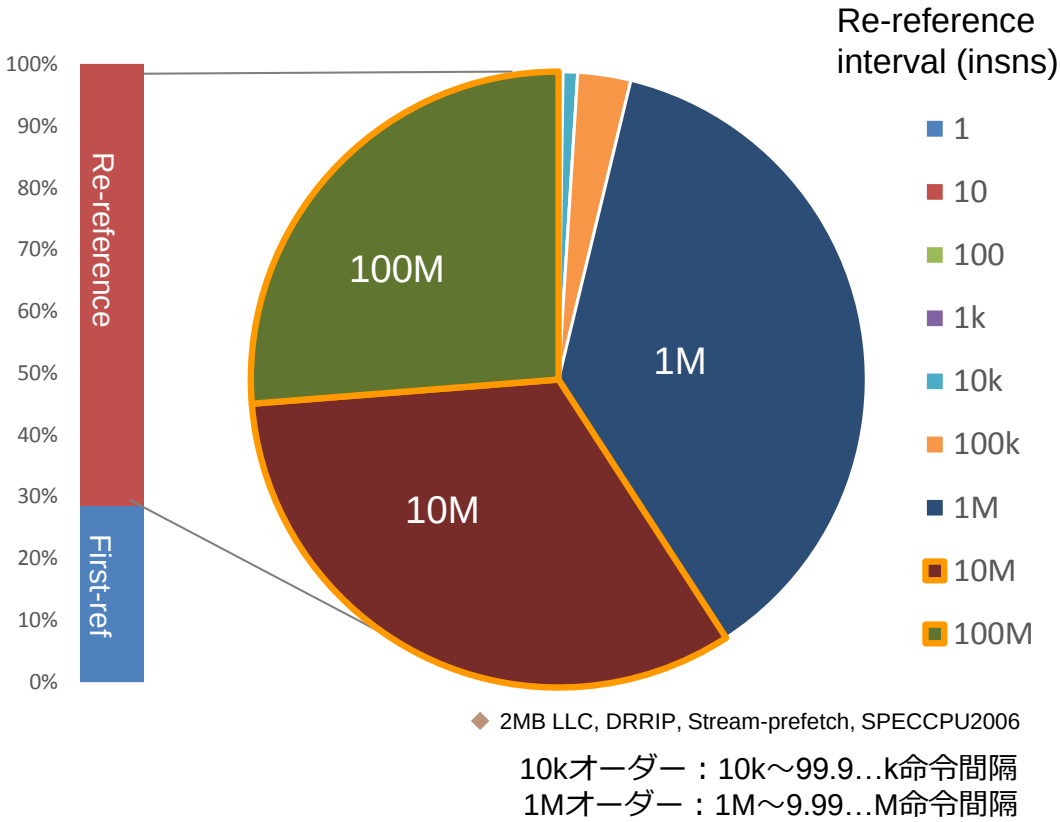



図 19 初期参照・再参照の割合と再参照距離別のキャッシュミスの割合

## 3.2. Stubborn 戦略の提案

### 3.2.1. アイデア

前節では残存するミスがどのようなものか調査した．ここまでの調査を踏まえ，なぜ，従来手法ではこのような長期再参照ミスが残るかについて，本研究では次のようにみなした．

- 従来の手法は直近の時間的空間的な傾向を利用して直後のアクセスを予想している
- しかし，長期再参照ミスを起こすようなアクセスパターンは，直近のアクセス履歴からは推測できない
- 長期に見て時間的な周期性のあるアクセスであっても，アドレスの空間的な連続性とアドレスの変移量を頼りにしているプリフェッチでは参照を予測することができないことがある

これに対して，本研究では長期にわたってラインを持ち続ける戦略があれば，従来手法では対処できなかった性質を持つミスに対処できるのではないかと考えた．これを踏まえ，直近の履歴に基づいた判断でキャッシュラインを保持できる程度の期間での再参照を保護するような従来の置き換えアルゴリズムに対して，新しいキャッシュアルゴリズムの基本戦略を提案する．その戦略は，より長い間隔での再参照の保護を目的として，ある基準で定めたキャッシュラインを，直近の履歴に影響されず，キャッシュから長期間追い出さないことを特徴する．本研究ではこれを Stubborn 戦略と命名する．この Stubborn 戦略は，キャッシュ中に追い出しを起こさない領域（Stubborn 領域）を設けることで実現させる．

本研究ではこの戦略を LRU，DRRIP の既存手法と融合し，ハイブリッドな構成とする．本研究における Stubborn 戦略の最も単純な実装では，LLC の最大半分を，Stubborn 戦略を適用する領域とする．LLC をすべて Stubborn 領域にしてしまわずに半分を従来手法のために残すのは，従来どおり短期の再参照によるヒットを維持し，またプリフェッチャが活用する領域を残すためである．

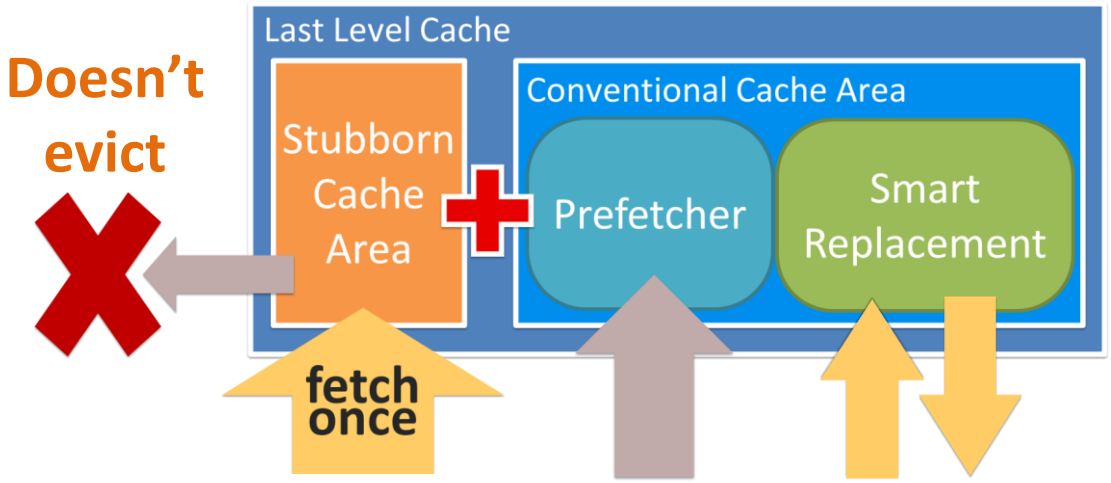



図 20 Stubborn 戦略の既存手法とのハイブリッド構成

ハードウェア上では，Stubborn 領域と既存手法による置き換えアルゴリズムは同一のセットに共存することになる．そのため，セット中の各ラインが追い出し不可の対象なのか追い出し可能であるかを判断するための機構が必要となる．ここで，Stubborn 戦略の判断により追い出さないとしたキャッシュラインには「追い出し不可属性」を付加する．この追い出し不可属性の付加は挿入時に行う．動作のモデルとして，LRU をベースとしたときの挿入・追い出し操作を図 21 に示す．

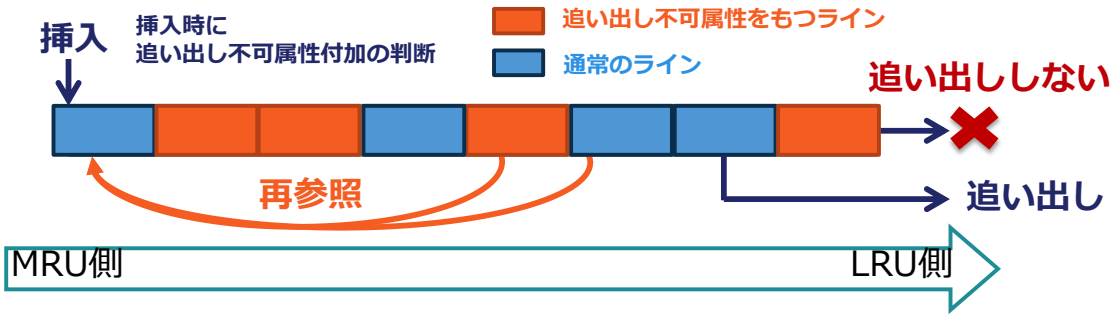



図 21 LRU based Stubborn における挿入・追い出しの操作

### 3.2.2. Normal Stubborn 実装

#### 3.2.2.1. キャッシュテーブル構成

Stubborn 戦略を既存手法と混合し，キャッシュテーブル上に Stubborn 戦略を実現するための方法として，それぞれのキャッシュラインに 1bit の Stubborn フラグを持たせ，追い出し不可属性とする．これを Stubborn フラグ bit と呼ぶ．挿入時の判断により Stubborn フラグが ON になったキャッシュラインは，一定の期間が経過するまで，あるいは全実行の終了までそのセット内で追い出しの対象とならない．置き換えアルゴリズムは追い出し時にこのフラグを参照し，追い出しの判断に用いる．このような機構の構成を次の図 22 に図示する．

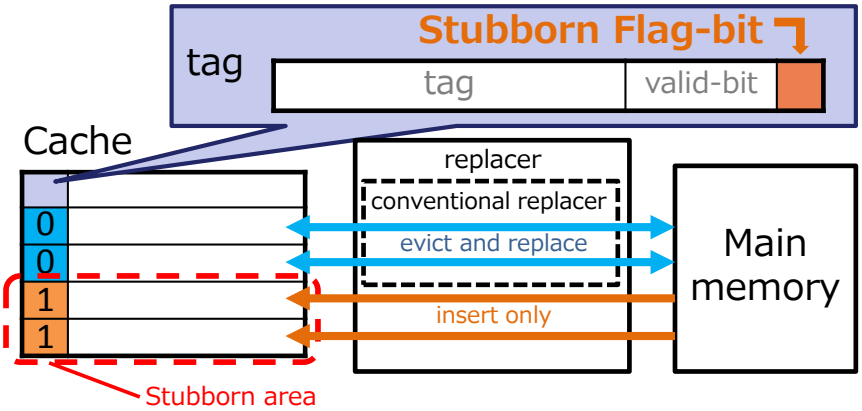



図 22 Stubborn 戦略のメモリサブシステム実装

#### 3.2.2.2. 追い出し不可属性の判断

ここでは追い出し不可属性のフラグを立てるかどうかの判断手法として，先着順採用を実装する．これは，セット中の Stubborn 領域が埋まるまでの期間，先に到着したキャッシュラインを置き換え不可属性に採用するというものである．図 23 にモデルを図示する．

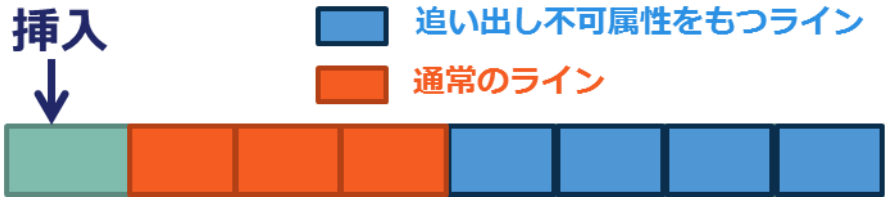



図 23 先着順採用

また，プリフェッチで挿入されたキャッシュラインについては，必要となるタイミングで再度プリフェッチにより挿入されることが予想されるためキャッシュに保護する必要性が低く，プリフェッチによる挿入では Stubborn フラグを立てないものとする．

### 3.2.2.3. 追い出し･挿入処理

Stubborn 戦略を適用したときの LRU ベース，DRRIP ベースの置き換えアルゴリズムにおける追い出し時と挿入時の処理を手順化して明記する．

追い出し時の操作については LRU ベースの場合と DRRIP ベースの場合で異なり，以下のような手順をとる．

**LRU ベースの Stubborn 領域を持つキャッシュの追い出し操作手順**

1. LRU オーダーの最も小さいラインの Stubborn フラグをチェック
2. Stubborn フラグが立っていれば次に LRU オーダーが小さいラインをチェック
3. Stubborn フラグが立っていなければそのラインを追い出す
4. 2 から繰り返す

**DRRIP ベースの Stubborn 領域を持つキャッシュの追い出し操作手順**

1. RRPV が最大で，かつ Stubborn フラグが立っていないラインを探す
2. 見つかればそのラインを追い出す
3. 見つからない場合はセット内の全てラインの RRPV をインクリメントする
4. 1 から繰り返す

これら追い出しライン選択手法は，セット中の全てのラインに Stubborn フラグが立っているということがないという前提で動作する．これは後述の挿入時の動作により保証される．

挿入時の操作は LRU ベース，DRRIP ベース共に共通で，以下のような手順をとる．

**挿入時の操作手順**

1. ベースの置き換えアルゴリズムが挿入先のラインを決定
2. 挿入先のセットの Stubborn フラグの立っているラインの数が way 数の半数未満なら，ラインに Stubborn フラグを立てて挿入

これらの操作によって，既存の置き換えアルゴリズムに付加する形で Stubborn 戦略が実現できる．

#### 3.2.2.4. Stubborn 領域の保持期間

Stubborn 領域の保持期間について, 本研究における Stubborn 戦略の最も単純な実装では 1G 命令実行の間，Stubborn 領域の入れ替えは起こさない．これは，10M～100M の再参照をターゲットとしているためである．1G 命令以上の期間を実行する実プロセッサ環境での適用の実現性について考えると，この手法は実行期間 1G 命令ごとに Stubborn フラグをまるまるクリアし，再び格納させるモデルを想定したときにこれと同等になる．

このような, 本研究における最も単純な実装である Stubborn 戦略の適用ポリシーを，以降の提案手法との区別のため Normal Stubborn と名付ける．また，LRU をベースとしたときの置き換えアルゴリズムを LRU based Normal Stubborn，DRRIP をベースとしたとき DRRIP based Normal Stubborn と呼ぶ.

### 3.2.3. 評価

上記の実装に基づいた置き換えアルゴリズム，LRU based Normal Stubborn および DRRIP based Normal Stubborn を評価した．1.4 節における評価環境をもとに，LLC の置き換えアルゴリズムを LRU, LRU based Normal Stubborn, DRRIP, DRRIP based Normal Stubborn とし，この 4 種で性能を比較する．

#### 3.2.3.1. IPC

IPC での評価を次の図 24 に示す. 縦軸は LRU の IPC を 1 としたときの相対 IPC の増減をパーセントで示している. LRU based Stubborn が LRU に対して, 最大で 26.1%の性能向上，幾何平均でも 1.0%の性能向上．内約として，483.xalancbmk で最も大きく 26.1%の伸びが得られ，次いで 471.omnetpp で 10.5%の性能向上が得られた．

DRRIP based Stubborn が DRRIP に対して，最大で 6.2%の性能向上，幾何平均でも 0.6%の性能向上．内約として，433.milc で最も大きく 6.2%の伸びが得られ，次いで 482.sphinx3 で 5.9%の性能向上が得られた．これら 4 手法の中では DRRIP based Stubborn が幾何平均で最良の結果を示した．

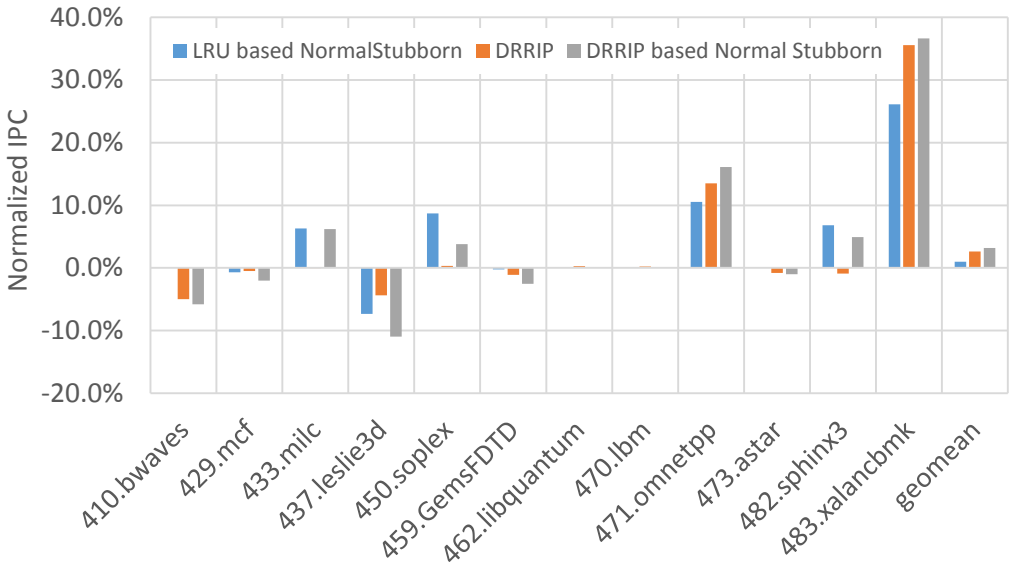



図 24 Normal Stubborn の IPC 評価

#### 3.2.3.2. 再参照距離別 MPKI

次に，ワークロードでの再参照距離別の MPKI での評価を図 25 に示す．縦軸は，再参照によるミス数を距離毎に分割し積み上げた MPKI で, 各ワークロードでは左から順に LRU, LRU-based Stubborn, DRRIP, DRRIP-based Stubborn を示す．列数が多いため 2 段のグラフとしているが，上下段での条件の差はない．

このグラフから，それぞれのワークロードにおいて，どのようなミスが削減できているのかを再参照距離の分類によって解析することができる．特に注目すべきは 433.milc, 450.soplex, 482.sphinx3 の再参照ミス距離の分布である．これらのワークロードでは，Stubborn 戦略の適用によって狙い通り残るミスのボリュームゾーンである 10M～100M オーダーの長期の再参照ミスを減らすことができている．これらは DRRIP でも対処できなかったミスであり，Stubborn 戦略によって初めて減らすことができたミスである．スラッシングにより 1M オーダーの再参照ミスを起こす 483.xalancbmk, 471.omnetpp では，Normal Stubborn により残るミスのボリュームゾーンである 1M 帯の再参照ミスを減らすことができているが，これについては DRRIP でも同程度の対処ができているために，DRRIP からの性能の伸びは小さく留まった．

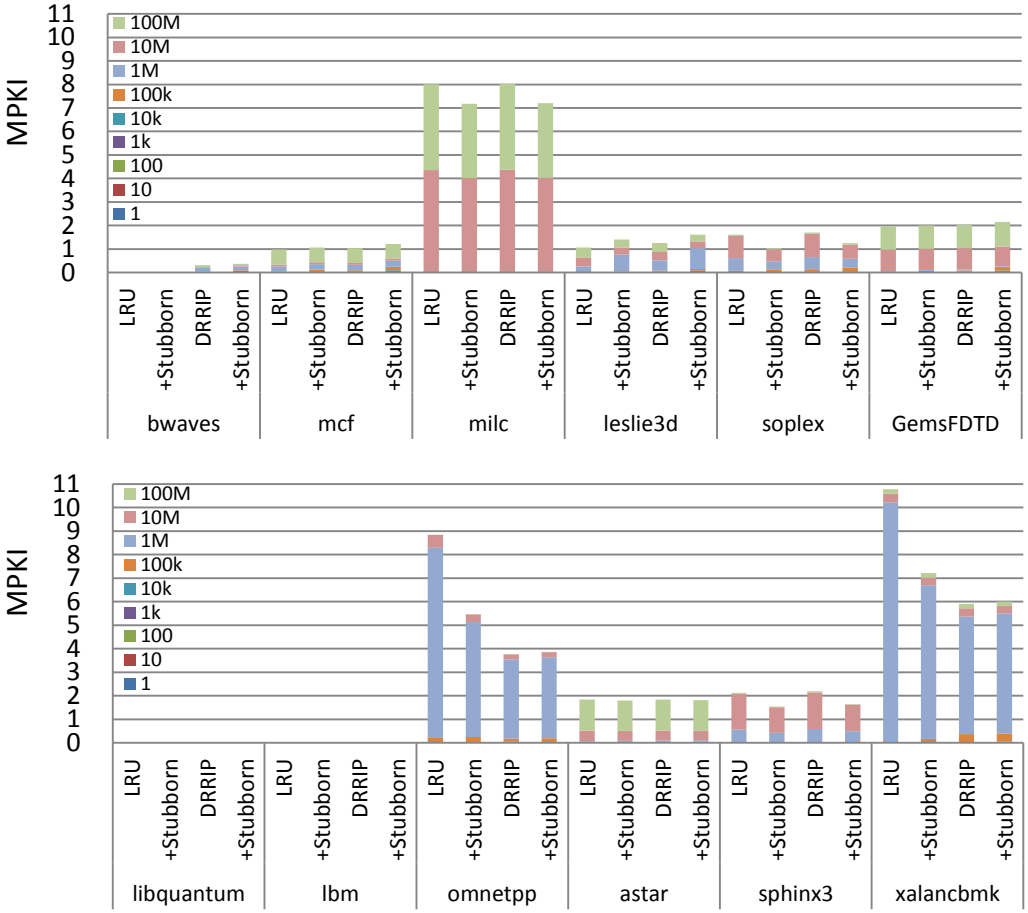



図 25 Normal Stubborn の MPKI 評価

この結果により，従来手法のような再参照予測では判断できなかった距離の再参照で生じていたミスを，Stubborn 戦略を適用した Normal Stubborn によりヒットさせることができ，長期の再参照アクセスが占める割合の高いワークロード，スラッシングが連続するワークロードで共に有効に機能することを確認した. 特に, 433.milc については, Newton らの研究[9]におけるデッドブロックに関する記述では 433.milc はスラッシングではなく，ほぼ 100%がデッドブロックとなるストリーミングアクセスであるとみなし，そのように報告されているが，Stubborn 戦略を採用した置き換えアルゴリズムではこれはデッドブロックではなく，キャッシュラインに保持することで対処できる再参照ヒット可能なスラッシングを生じるアクセス群であることを示した.

#### 3.2.3.3. アクセスパターンのプロットで見る Stubborn 戦略の効果

前節では，Stubborn 戦略の添加により従来手法のような再参照予測では判断できなかった 1～100M オーダーの再参照によるミスを防ぐことができていることを示した．ここでは，そのミス削減の様子を，キャッシュアクセスのログをプロットし図示したグラフを用いて具体的に確認する．

一例として，483.xalancbmk における DRRIP と DRRIP based Stubborn でのアクセスを次の図 26 に図示する．この結果では，Stubborn 戦略の特性が良く表れている．DRRIP においてはヒットが続くアドレス領域が離散しており，また局所的に時間的局所性が強い箇所でのヒットが見られるが，一方で DRRIP based Stubborn では実行開始直後に使用した領域が Stubborn 領域に固定されることで，明確に決まったアドレス帯でのヒットが続いている．また，これにより 483.xalancbmk における 1M 命令程度の間隔で繰り返される激しいスラッシングに対する耐性を持っている．

この結果の図示では，先着順で採用する Stubborn 戦略の特性によりヒットする領域が固まっていることで DRRIP based Stubborn でのヒット数が DRRIP より視覚的に極端に多くみえるが，実際は図 25 に示したように同程度のミス数である．

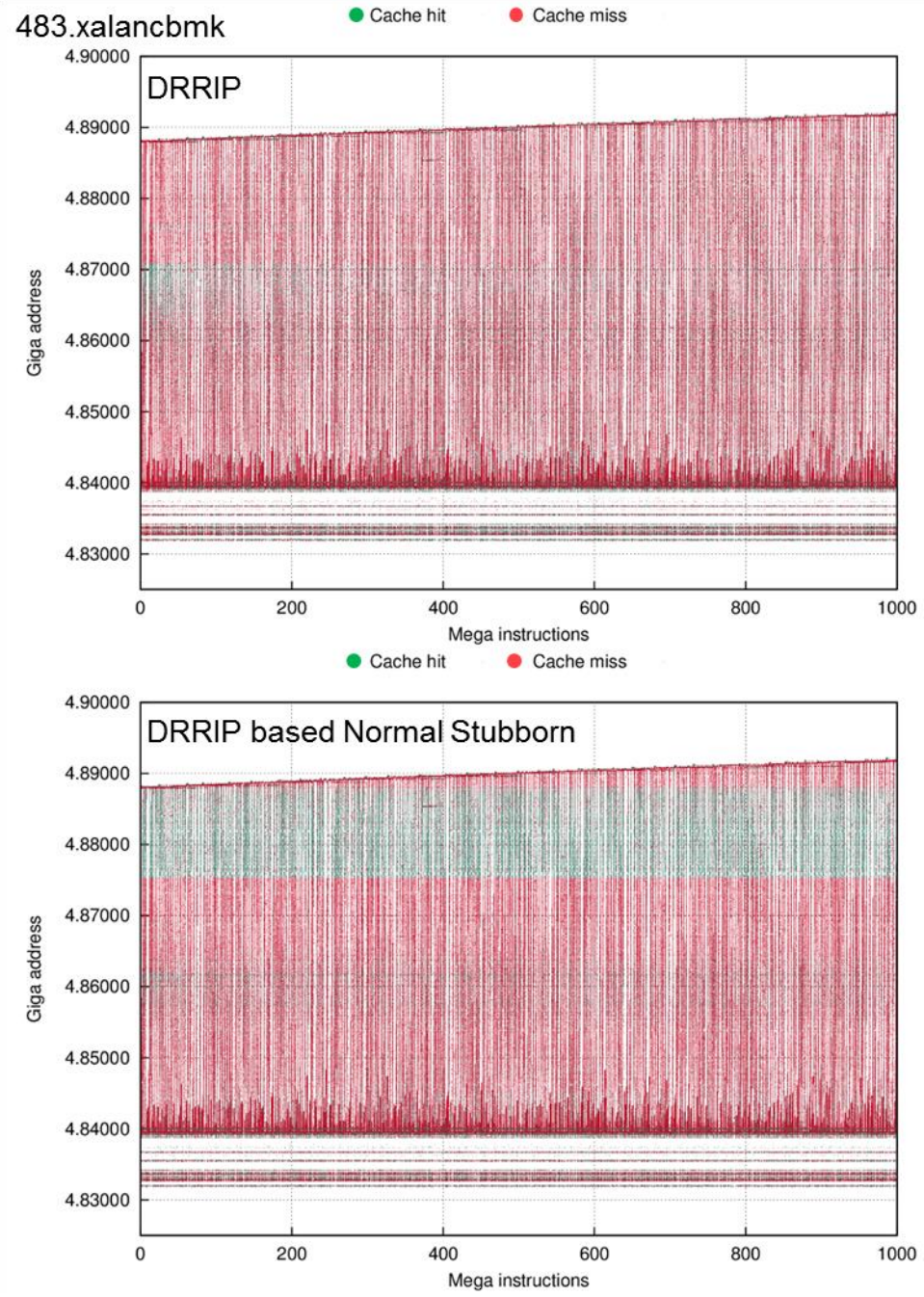



図 26 483.xalancbmk におけるアクセスパターンのプロット

次に，482.sphinx3 での DRRIP と LRU based Stubborn でのアクセスを次の図 27 に図示する．この結果では，ヒットが集中しているアドレス帯がわかりやすく，抽出しやすいため別途図示した．482.sphinx3 では 25M 命令程度の間隔でのスラッシングにより生じる再参照ミスが残るミスのボリュームゾーンであり，DRRIP においても最も長いアドレス帯でのスラッシングアクセスはヒットさせられているが，LRU based Stubborn においては同様のアドレス帯でのヒットに加え，実行開始直後で参照した他のアドレスでのスラッシングアクセスについても，ワークロード全体でのスラッシング量に惑わされずに保持し，これをヒットさせることができている．この差分が，LRU based Stubborn ，DRRIP based Stubborn で達成することができた 10M オーダーの再参照ミスの削減を示している．

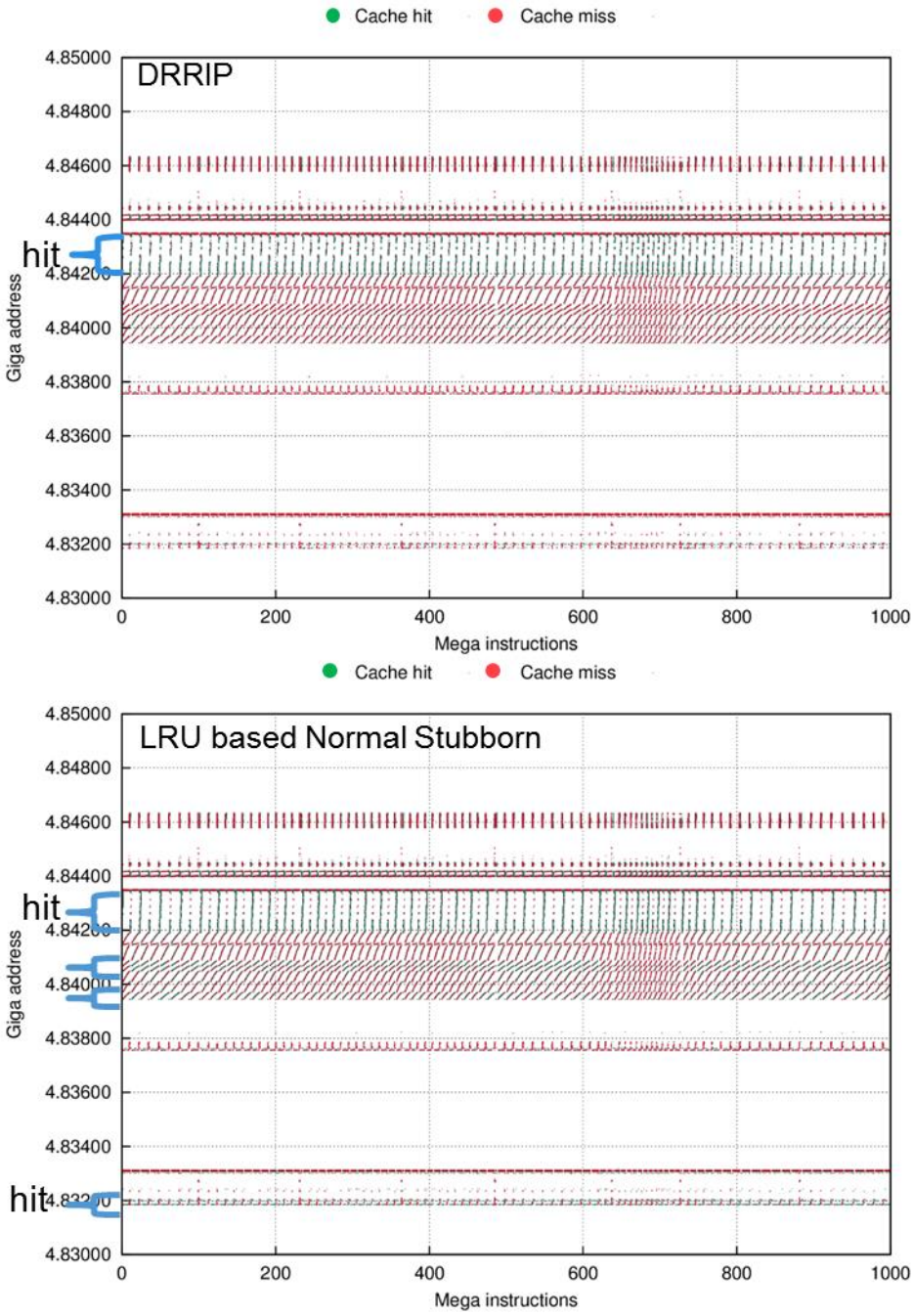



図 27 482.sphinx3 におけるアクセスパターンのプロット

### 3.2.4. スラッシング耐性についての考察

前節で確認したような Stubborn 戦略によって得られたスラッシング耐性について，モデルを立ててその効果が得られる理由について具体的な挙動を考え，従来手法と比較する．

モデルとして，最もシンプルなスラッシングを想定する．このモデルでは，単一セットに対してアクセスを生じるラインを共有しないアドレスへのアクセスのみプロットし，各置き換えアルゴリズムの動作をトレースする．モデルにおける 1 セットあたりの way 数，スラッシング周期は条件毎に提示する．

#### 3.2.4.1. LRU のスラッシング耐性

まず，LRU におけるスラッシング耐性について考える．LRU では，スラッシングの一周のアクセス数が way 数+1 を越えた時点で以降全てのアクセスでミスとなる．これは，LRU における挿入時の LRU オーダーが固定であることで，追い出し時の判断が全て挿入順に依存してしまうためである．また，SRRIP においても，RRPV が最大値となった追い出し候補が複数あるときに追い出しの選択にランダム性を持たせずに順に追い出した場合は同じ理由で LRU と同様の結果になる．

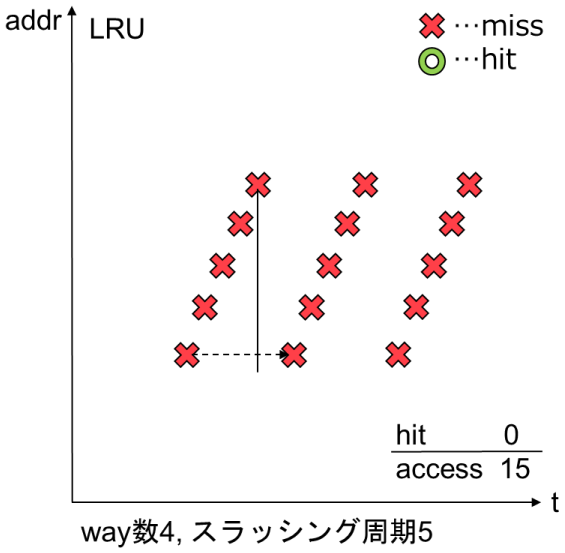



図 28 LRU のスラッシング耐性

### 3.2.4.2. BRRIP のスラッシング耐性

BRRIP においては，スラッシング耐性を持たせるための工夫として挿入時にランダムに相対的に優遇するラインを発生させることで，RRPV が最大値になるまでのタイムラグを生じさせ，追い出し時の判断が全て挿入順に依存してしまうことを防いでいる．

図 29 の左側に示すように，BRRIP では SRRIP と同様にスラッシングによるミスが連続している状態で，一定の確率で RRPV を最大値で挿入されたラインがあった際，挿入順とは別にこれが優先して追い出されるため 1 つのラインがヒットするようになり RRPV が 0 でリセットされる．以降，RRPV が 0 となったラインは追い出されることがなくキャッシュラインに残ることができる．

しかし，このように BRRIP がスラッシングに耐性を持つのは 1 周のアクセスが way 数の 2 倍のスラッシングまでで，図 29 の右側に示すように way 数の 2 倍の数+1 を越えた時点で以降全てのアクセスでミスとなる．このように，スラッシング耐性を意識した BRRIP をもってしても，一定以上のスラッシング周期で全てのアクセスがミスとなってしまう．

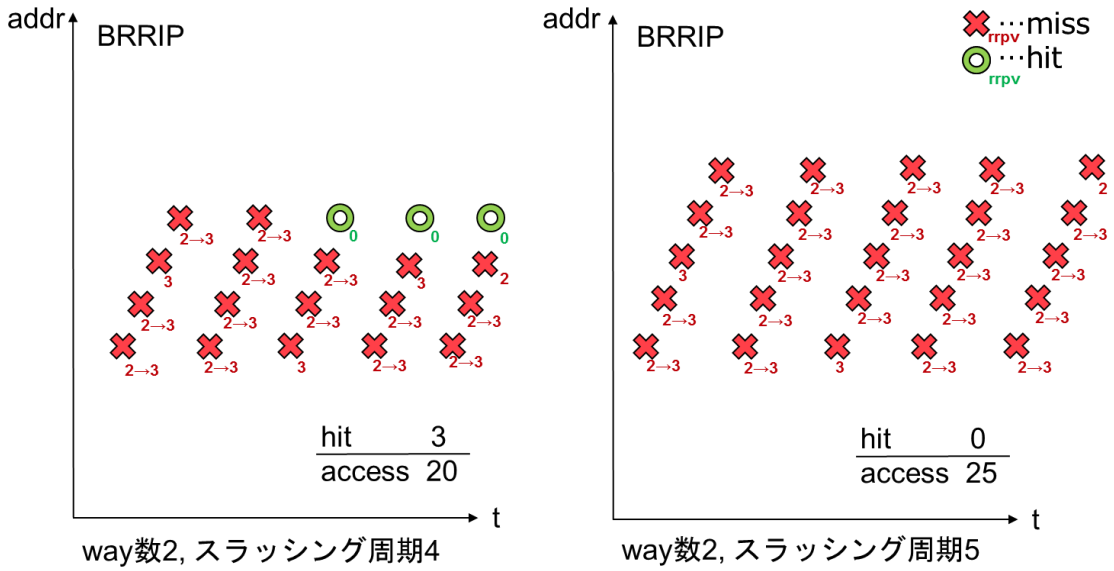



図 29 BRRIP のスラッシング耐性

### 3.2.4.3. Stubborn 戦略のスラッシング耐性

Stubborn 戦略を適用したポリシーでは，あるキャッシュラインを追い出さないという戦略そのものがスラッシング耐性を持たせるための工夫となる．次の図 30 に BRRIP based Stubborn でのアクセスパターンを示す．BRRIP では way 数の 2 倍の周期のスラッシングが限界であったが，Stubborn 戦略では 2 倍+1 を越える周期のスラッシングであっても way 数の半分はヒットが続くことになる．

また，このアクセスが連続している状態で，RRPV が 0 となったキャッシュラインがもう一つ現われれば，そのラインも RRIP の追い出しポリシーに基づいて追い出しされなくなる．これにより，このケースではセット中の 3/4 のキャッシュラインがヒットし続けることになる．

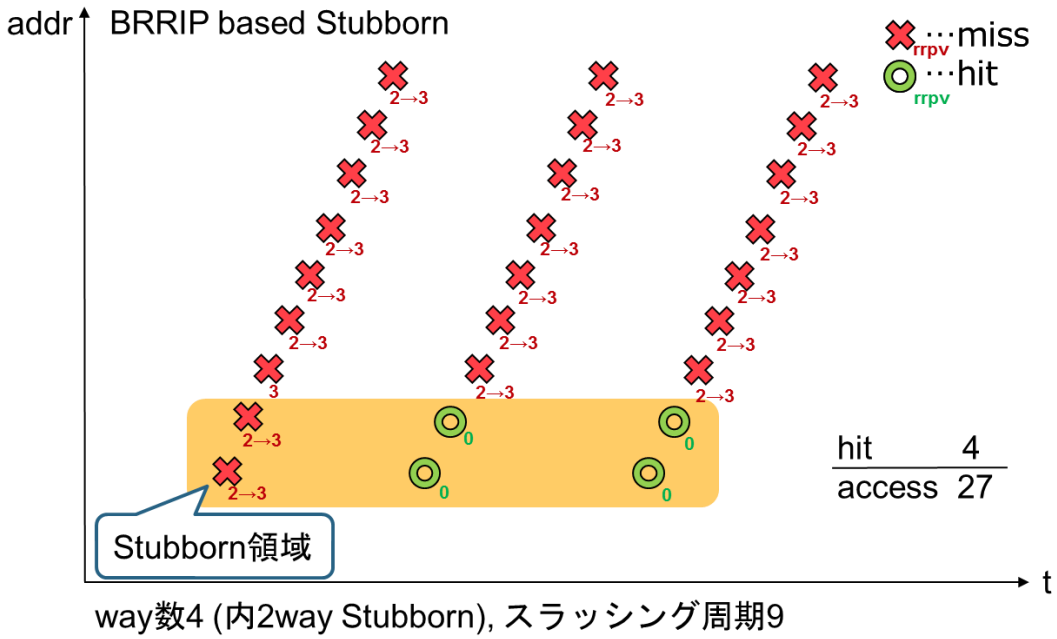



図 30 BRRIP based Stubborn のスラッシング耐性

これらのスラッシング耐性の例示はあくまで 1 セットに対する単純なスラッシングによるもので，実際のワークロードでは再参照があるアドレスへのアクセスとないアドレスへのアクセスが入り交じる．アクセスするアドレス帯が変動するなど，複雑な条件となるので Stubborn が常に従来手法より優れた結果を示すわけではないが，全体に占めるスラッシングアクセスの割合が大きい 483.xalancbmk などにおいてはこのモデルでの動作に近い挙動となっていることが確認できる．

また，Stubborn 戦略では，スラッシング耐性と同時にストリーミングアクセス，ランダムアクセス混在のアクセスパターンへの耐性ももつ．図 31 に示すように，再参照されることのない沢山のランダムアクセスと，長期の再参照距離ではあるが再参照があるアクセスが混在しているケースでは，再参照があるキャッシュラインを Stubborn 領域で保持することができればこれをヒットさせ続けることができる．

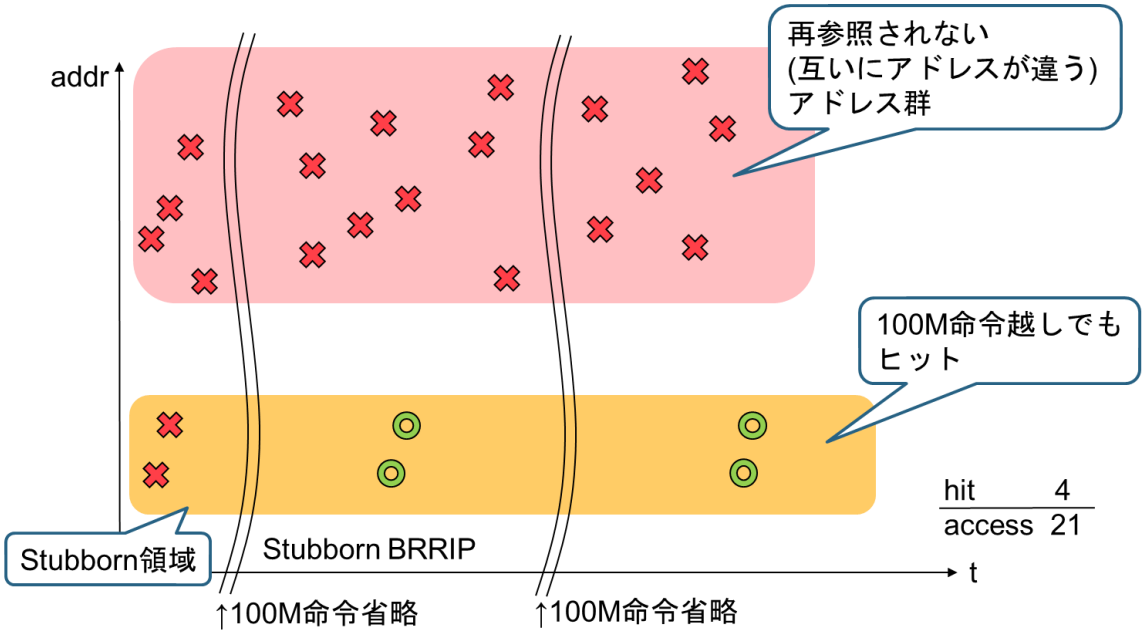



図 31 Stubborn 戦略のランダムアクセス混在パターン耐性

## 3.2.5. まとめ

本研究による調査で, LLC で生じる残されたキャッシュミスの多くを占めるものが再参照ミスであり，また 10～100M オーダーに渡る長期の命令数間隔に渡る再参照であることを明らかにした．これに対処するため，本研究ではより長い間隔での再参照の保護を目的とした Stubborn 戦略を提案した．これは，キャッシュライン追い出しを一時的に凍結し，長期に渡って追い出さないことで長期保持による統計的なヒット数向上を狙い，またスラッシング耐性を持たせることを狙いとしたキャッシュライン追い出し対象選択の戦略である．Stubborn 戦略は，これまでの履歴ベースの戦略ではリーチしなかった長期再参照をカバーする．

この Stubborn 戦略は関連研究と競合するものではなく直交する手法であり，置き換えアルゴリズムについては LRU や DRRIP のようにベースラインとして組み込んで共存することができる．

本章では，本研究における Stubborn 戦略の最も単純な実装である Normal Stubborn を評価した．結果より，従来手法のような再参照予測では判断できなかった距離の再参照で生じていたミスを，Stubborn 戦略を適用した Normal Stubborn によりヒットさせることができ，長期の再参照アクセスが占める割合の高いワークロード，スラッシングが連続するワークロードで共に有効に機能することを確認した．さらに，従来手法のような再参照予測では判断できなかった距離の再参照で生じていたミスは，周期の長いスラッシングによってもたらされていることを明らかにした．

Normal Stubborn の実装では，どのラインを追い出し不可とするかの判断として，先着順で挿入されたものを採用している．どのキャッシュラインを Stubborn 領域に入れて保持し続けるかの選択手法は，Stubborn 戦略にとって当然重要であるが，履歴をみても当てられないような長期再参照と，履歴をあてにすると追い出されてしまうスラッシングに対策するために採択した履歴を使わない方法として，この先着順採用での選択は一定の効果を示した．

このようなシンプルな戦略と機構，実装にもかかわらず性能向上が得られている．しかしながら，Stubborn 戦略はワークロードの特性によって効果が大きく変化する．従来のどの手法でも対応できなかったワークロードでミスとなるようなケースでその戦略により長期再参照をヒットさせ性能向上が得られる一方で，437.leslie3d での性能低下が示すように，プリフェッチの恩恵が強く受けられるワークロードでの副作用が大きい．無駄なラインを保持し続ければ，キャッシュの容量効率を低下させてしまう．

# 第4章 Stubborn 戦略のポテンシャル調査

Stubborn 戦略を適用したナイーブな実装である Normal Stubborn ではシンプルな戦略と機構，実装にもかかわらず性能向上が得られた一方で，副作用の発生も確認された．このため，Stubborn 戦略の組み込みにはどのキャッシュラインをどれだけの期間 Stubborn 戦略によって固定するかの決定が欠かせない．そこで，本章では， Stubborn 戦略が有効に機能する条件について複数のアプローチにより検証，調査することで，さらなる性能向上を得つつ副作用を抑制するための条件を明らかにし，Stubborn 戦略のポテンシャルを引き出す方法を明らかにする．

## 4.1. キャッシュ容量と Stubborn 戦略の関係

### 4.1.1. 検証

Stubborn 戦略の効果はワークロードのワーキングセット（データサイズ）とキャッシュ容量の関係に依存することが予見される．3.2.4 節で示したように，Stubborn 戦略がスラッシング耐性として機能する際に，コンスタントに保持して再参照させられるラインの割合はスラッシングの周期に依存する．キャッシュ容量が半分になれば単位時間に 1 つのセットに生じるアクセスの数と参照対象のアドレスの数は単純には倍となり，その影響により Stubborn 戦略で固定していれば得られたはずのヒット率の割合も半分になる．また，キャッシュ容量の変化はプリフェッチとの共存にも影響する．3.2.3 節での Normal Stubborn の評価において評価全体ではトータルで性能向上が得られているのは，半分の Stubborn 領域と半分の従来手法のための領域という構成，プリフェッチと共存してバランスしているためであるが，これはキャッシュ容量に依存しているのか，キャッシュ容量の増減に関わらず機能するのか，という疑問が生じる．これについて，本節における検証で明らかにする．

### 4.1.2. 検証方法

Normal Stubborn の実行において，LLC 容量を 256KB, 512KB, 1MB, 2MB, 4MB, 8MB と設定した環境でワークロードを実行し，評価する．ここで，キャッシュ容量に関わらずセットあたりの way 数は 8 way, Stubborn 領域はその半分の 4way で固定する．また，LLC のレイテンシ，L1，L2 キャッシュの容量も表 1 の通り固定とする．

### 4.1.3. 評価

キャッシュ容量変化による Stubborn 戦略特性の傾向を図 32 に示す．ここで，縦軸は IPC，横軸は LLC キャッシュ容量でプロットしている．表記の省略のため，LRU based Normal Stubborn は StubbornLRU，DRRIP based Normal Stubborn は StubbornDRRIP と表記する．

この図示により，それぞれのワークロードごとに Stubborn 戦略の適用がどの程度のキャッシュ容量から有効となり，どの程度の容量になると各ポリシーの動作に関係なく性能が収束していくのかがわかる．容量に関わらず一貫して Stubborn 戦略の適用が副作用を生じるケースも可視化されている．

分類すると，まず Stubborn 戦略が効果的に機能している（具体的には，IPC が LRU 比で 5%以上向上している）ワークロードとその容量範囲は

- 471.omnetpp : 512KB～2MB
- 483.xalancbmk : 1MB～4MB
- 482.sphinx3 : 1MB～8MB
- 433.milc : 1MB～8MB
- 450.soplex : 2MB～8MB

といった分布となった．

次に，一定容量になると各ポリシーの動作に関係なく性能が収束するワークロードとその収束時の容量は

- 429.mcf : 8MB
- 437.leslie3d : 8MB
- 459.GemsFDTD : 4MB
- 462.libquantum : 256KB
- 470.lbm : 512KB
- 471.omnetpp : 8MB
- 483.xalancbmk : 8MB

という分布となり，これらのワークロードではこの容量に達するとキャッシュ容量がワーキングセットサイズを満たし，また並行する多数のストリーミングアクセスがある場合でもプリフェッチで対応できていることがわかる．

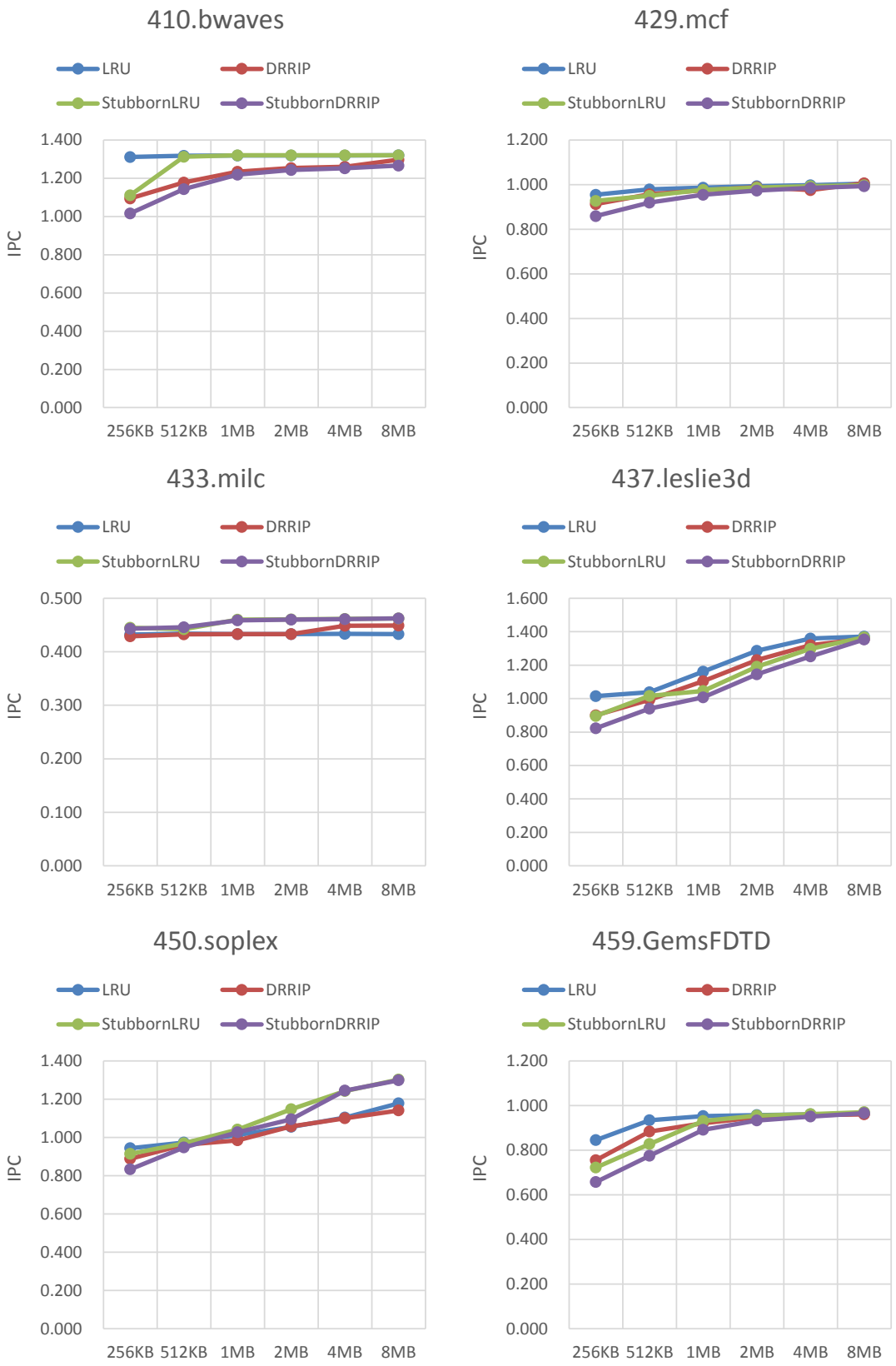
これに対して，先に挙げた Stubborn 戦略の適用が効果的なワークロードのうち，482.sphinx3，433.milc，450.soplex では 8MB の容量を持ってしてもキャッシュ容量がワーキングセットサイズを充足しない，アクセスパターンもプリフェッチでは対応できないワークロードとして認識できる．この 3 つのワークロードは共通して周期が長いスラッシングのアクセスパターンの傾向を持つ．

256KB，512KB の低容量帯で Stubborn 戦略が従来手法を阻害する（IPC が LRU 比で 5%以上低下している）のは次の

- 410.bwaves
- 429.mcf
- 437.leslie3d
- 450.soplex
- 459.GemsFDTD
- 473.astar
- 483.xalancbmk

のケースである．これらのワークロードでは 512KB 以下の低容量な環境では Stubborn 戦略を適用すべきでない．特に，2MB 構成の Normal Stubborn の評価において大きな性能向上が得られた 483.xalancbmk についても 256KB 構成では 10%以上の性能低下となっており，Stubborn 戦略適用の是非はキャッシュ容量とワーキングセットのサイズの比に大きく依存することがわかった．考察として，ワークロードによって Stubborn が効く容量帯に違いがでるのは，ほぼスラッシングの一周の長さに依存しているのではないかという推測ができる．

容量に関わらず一貫して Stubborn 戦略の適用が副作用を生じるのは 437.leslie3d のケースで，このワークロードについてはキャッシュ容量の多寡に関わらず，Normal Stubborn は適用すべきでない．


410.bwaves
LRU
DRRIP
StubbornLRU
StubbornDRRIP
IPC
1.400
1.200
1.000
0.800
0.600
0.400
0.200
0.000
256KB 512KB 1MB 2MB 4MB 8MB
429.mcf
LRU
DRRIP
StubbornLRU
StubbornDRRIP
IPC
1.200
1.000
0.800
0.600
0.400
0.200
0.000
256KB 512KB 1MB 2MB 4MB 8MB
433.milc
LRU
DRRIP
StubbornLRU
StubbornDRRIP
IPC
0.500
0.400
0.300
0.200
0.100
0.000
256KB 512KB 1MB 2MB 4MB 8MB
437.leslie3d
LRU
DRRIP
StubbornLRU
StubbornDRRIP
IPC
1.600
1.400
1.200
1.000
0.800
0.600
0.400
0.200
0.000
256KB 512KB 1MB 2MB 4MB 8MB
450.soplex
LRU
DRRIP
StubbornLRU
StubbornDRRIP
IPC
1.400
1.200
1.000
0.800
0.600
0.400
0.200
0.000
256KB 512KB 1MB 2MB 4MB 8MB
459.GemsFDTD
LRU
DRRIP
StubbornLRU
StubbornDRRIP
IPC
1.200
1.000
0.800
0.600
0.400
0.200
0.000
256KB 512KB 1MB 2MB 4MB 8MB

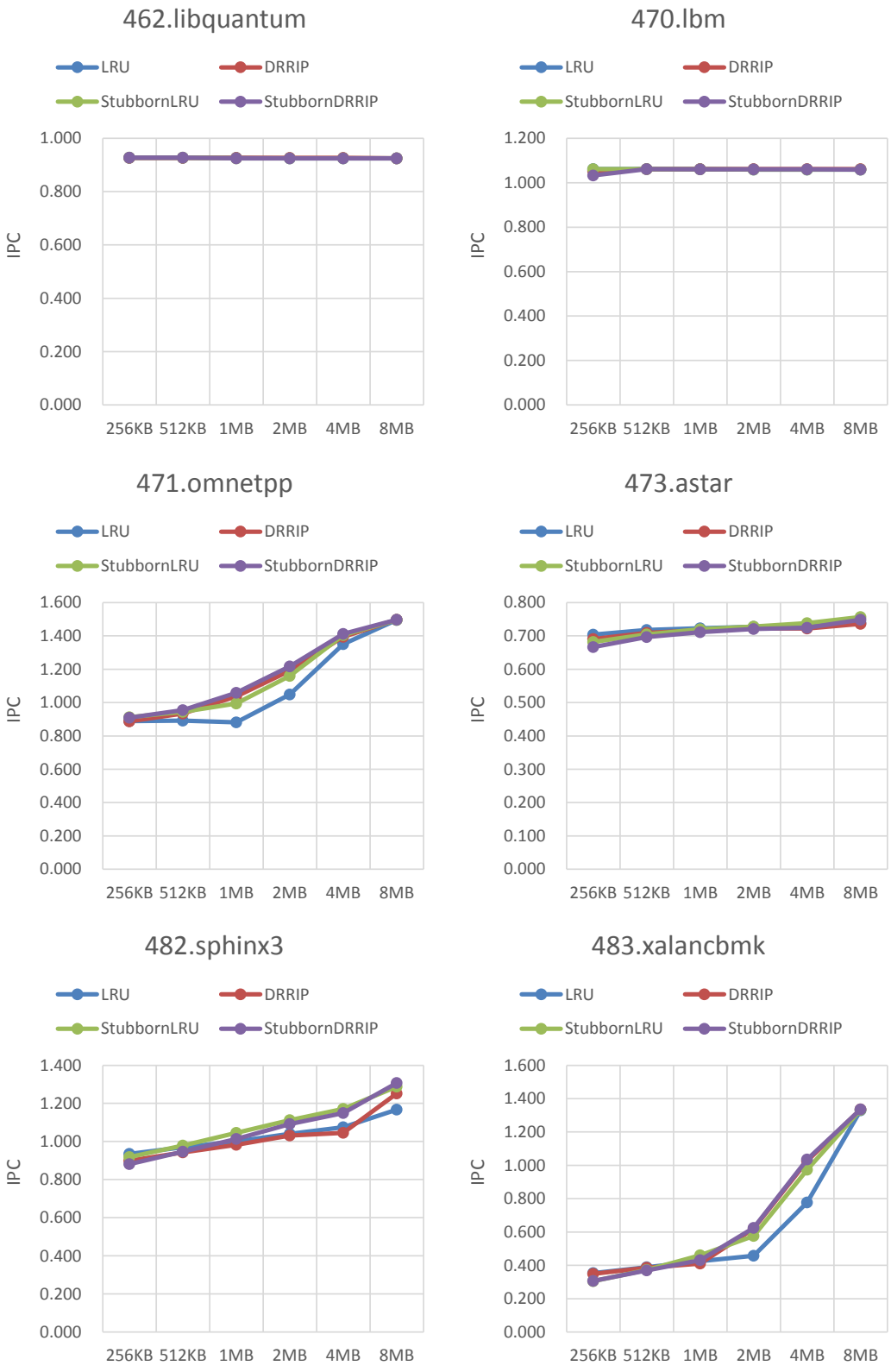



図 32 キャッシュ容量変化による Stubborn 戦略特性の傾向

### 4.1.4. まとめ

本検証では，キャッシュ容量の変動にともない Stubborn 戦略の適用が有効となる範囲について明らかにした．また，一定以上のキャッシュ容量で Stubborn 戦略がより強く有効となるワークロード，キャッシュ容量の大きさに関わらず Stubborn 戦略を適用すべきでないワークロードについても確認できた．この知見は 4.2 節以降の Stubborn 戦略の適応的な使用における判断基準を下すための参考となる．

## 4.2. Stubborn 戦略のフェーズ適応

### 4.2.1. アイデア

Normal Stubborn の結果より，Stubborn 戦略の適用では性能低下を生じるケースがあった．例えば，437.leslie3d では Normal Stubborn の適用による性能低下が大きく，これは 437.leslie3d がストリーミングプリフェッチの恩恵を強く受けられるようなアクセスパターンを持っているためである．このような，同時並行する複数のストリーミングアクセスを持ち，プリフェッチによりヒットさせることができるワークロードでは Normal Stubborn によるキャッシュ領域の半分の占領がプリフェッチを阻害することになる．このようなワークロードでは，プリフェッチを促進すべきタイミングでは Stubborn 戦略を適用すべきでない．

一方で，図 33 に示すように，実行フェーズによっては LLC ヒット率が低い期間があり，このような期間ではプリフェッチをもってしてもキャッシュミスを減らすことができず，また同時並行するストリーミングのためのプリフェッチにより再参照のあるラインが追い出されている．

つまり，ワークロードにはプリフェッチが強く有効であるため Stubborn 戦略を適用するべきでないタイミングと，適用することで従来手法では対応できなかったミスを減らすことができるタイミングがある．そこで，Stubborn 戦略の適用を，フェーズを基準として ON/OFF する手法を発案し，効果を調査した．

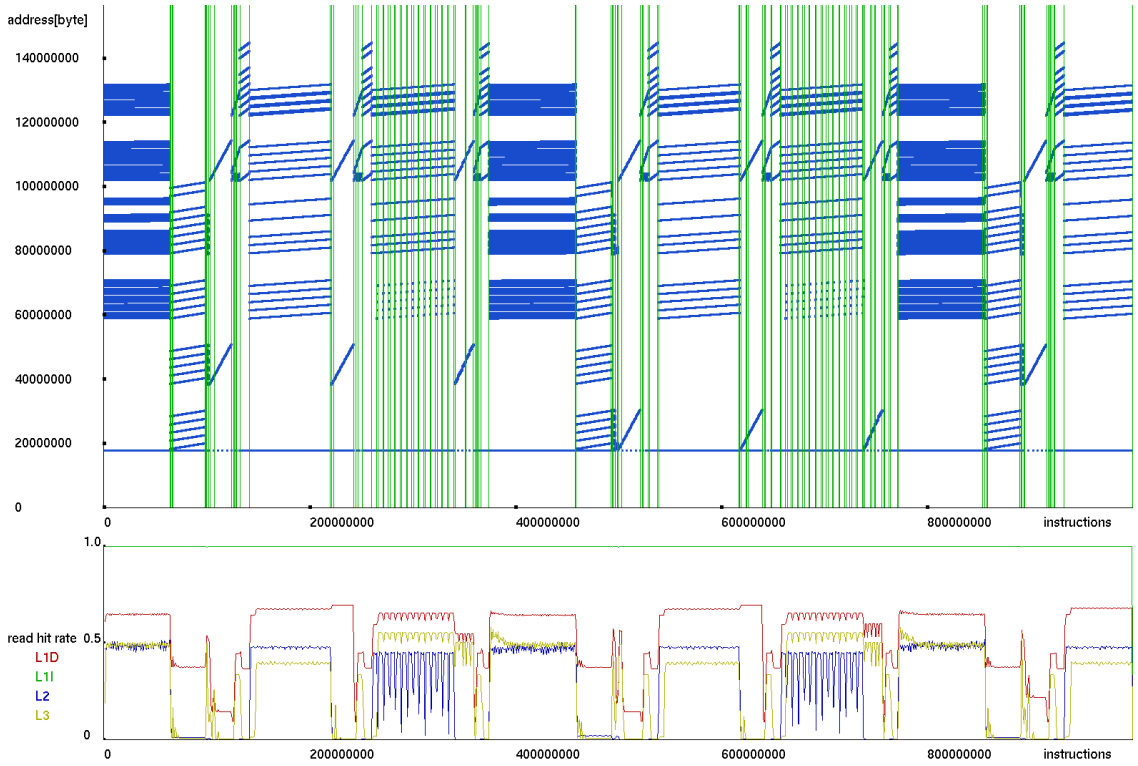



図 33 フェーズビューワ[37]による 437.leslie3d でのアクセスパターン解析

## 4.2.2. 調査方法

調査方法として，まずプリフェッチの恩恵が顕著であり，またフェーズごとの LLC ヒット率の差が顕著である 437.leslie3d において，フェーズに合わせた Stubborn 戦略の適用の切り替えが有効であるかどうかを調べる．

Stubborn キャッシュに占める Stubborn 領域の割合を 1 セットあたりの way 数で指定し，命令数を基準とした時間範囲で指定できる調査用の置き換えアルゴリズム Stubborn-Manually を実装し，評価した．Stubborn 戦略適用 way 数は 0 から 7 の範囲で指定できる．Stubborn 戦略適用のステータスの切り替えコマンドとして，OFF, ON, RESET を用意する．OFF は Stubborn 戦略を適用しない，ON は Stubborn 戦略を適用する，RESET はその時点でキャッシュラインに残っている Stubborn フラグをクリアしたのち Stubborn 戦略を再度適用 （ON）する，という動作指定を意味する．

ここで，Stubborn 領域以外の残りの領域を LRU で制御するものを LRU based Stubborn-Manually，DRRIP で制御するものを DRRIP based Stubborn-Manually とする．

図 33 のフェーズ変動を元に，437.leslie3d における Stubborn 戦略 ON/OFF 切り替えタイミングをピックアップし，次の表 2 のように指定した．ここで，命令数とステータスと合わせて指定する適用 way 数については，Stubborn 領域の常時 4 way 割り当てを行う Normal Stubborn との比較のために 4 way での指定と，合わせて 0 から 7 までの 8 つのパターンで実行ごとに固定して指定した．

表 2 437.leslie3d でのフェーズ適応のための Stubborn-Manually コマンド指定

| insns | status |
|---|---|
| 0 | OFF |
| 65710000 | RESET |
| 140690000 | OFF |
| 220720000 | RESET |
| 260320000 | OFF |
| 360570000 | RESET |
| 374050000 | OFF |
| 459140000 | RESET |
| 537490000 | OFF |
| 618370000 | RESET |
| 657120000 | OFF |
| 758210000 | RESET |
| 770850000 | OFF |
| 854250000 | RESET |
| 932600000 | OFF |

### 4.2.3. 評価

この調査の評価結果を次の図 34, 図 35 に示す. それぞれベースラインとする LRU, DRRIP での IPC を 1 としたときの相対 IPC の増減をパーセントで示している. 比較対照はベースライン，Normal Stubborn，フェーズ適応させた Stubborn-Manually である．この結果より，Normal Stubborn での Stubborn 領域の常時 4 way 割り当てと，フェーズによる切り替えを行ったときの 4 way 割り当てを比較して，LRU ベースライン，DRRIP ベースライン共に Stubborn 戦略を適用するタイミングの制御により Normal Stubborn で生じていた性能低下を抑えることができていることがわかる．また，LRU ベースラインでは 1 way，2 way 指定のとき 0.2%とわずかながら LRU に勝る性能が得られた. これは, プリフェッチを阻害することがない期間に, 適当な適用 way 数の選択をすることができれば，部分的にでも Stubborn 戦略を適用することでミスが削減できるタイミングがあることを示している.

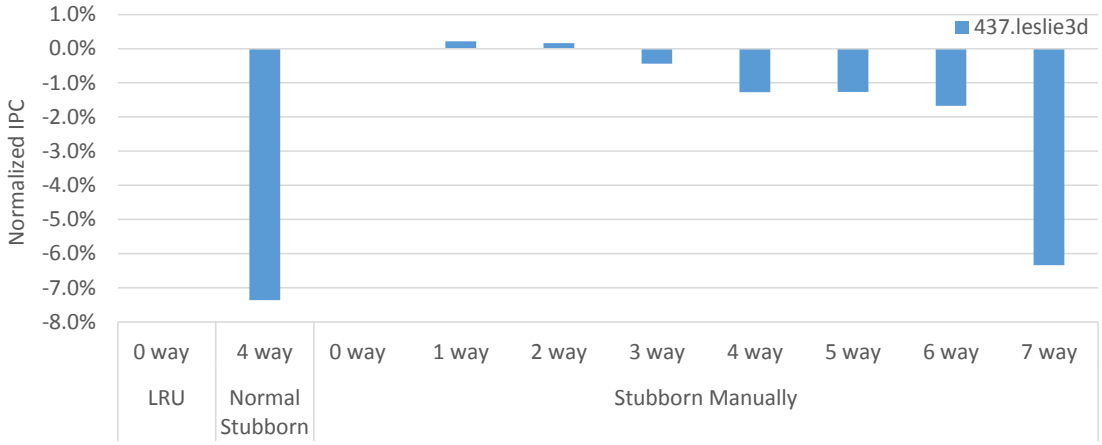



図 34 フェーズに適応した Stubborn-Manually の 437.leslie3d への適用結果
（LRU ベースライン）

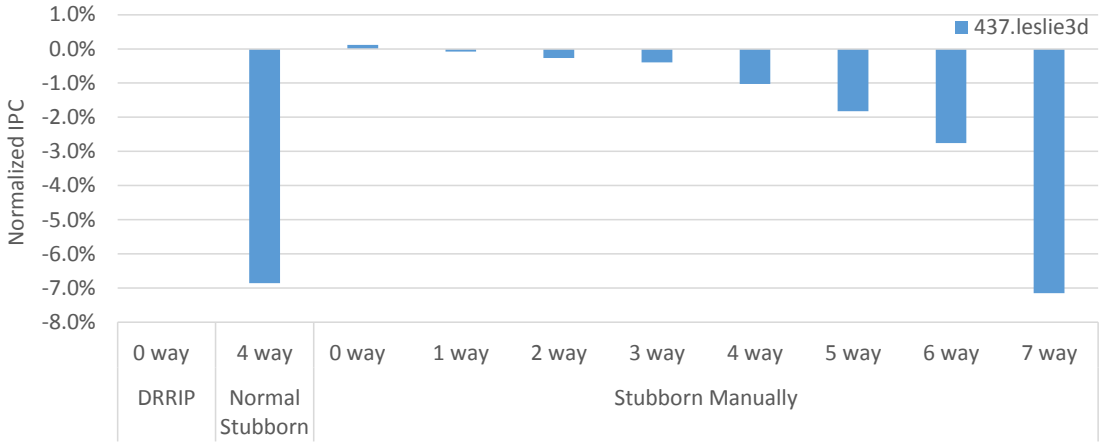



図 35 フェーズに適応した Stubborn-Manually の 437.leslie3d への適用結果
（DRRIP ベースライン）

### 4.2.4. まとめ

本調査では，Stubborn 戦略の適用による副作用を抑えるためには，フェーズに合わせた Stubborn 戦略の適用の ON/OFF 切り替えが有効であることを確認した．

しかしながら，フェーズの情報を用いた Stubborn 戦略の適用はこの調査のようにそれぞれのワークロードごとにフェーズの情報があらかじめ必要となるため，現実的ではない．この Stubborn 戦略適用の判断を実行時に行い，動的に ON/OFF の切り替えを判断できれば Stubborn 戦略のより実用的な拡張となると予想した．この結果を踏まえ，第 5 章では，Stubborn 領域の動的な ON/OFF 切り替えにより Stubborn 戦略の副作用の抑制とさらなる性能向上を図る．

## 4.3. 静的な Stubborn 領域の増減

### 4.3.1. アイデア

Normal Stubborn における Stubborn 戦略の実装では, LLC 容量の半分を従来手法, 残りの半分を Stubborn 戦略に基づく置き換えアルゴリズムで使用していた. LLC をすべて Stubborn 領域にしてしまわずに，半分を従来手法のために残すのは，従来どおり短期の再参照ミスを防ぎ，またプリフェッチャが活用する領域を残すためである.

しかし，適用 way 数を半数に固定することが最適とは限らない. ワークロードやフェーズごとに最適な Stubborn 戦略の適用度合いの強さがあると予想される. そこで，本調査では，LLC に占める Stubborn 領域の割合を変化させ，ワークロードごとにそれぞれ Stubborn 領域がどの程度の割合を占めるのが最適であるのかを調査した.

Stubborn 戦略の適用度合いの強さは，セットあたりの Stubborn フラグを立てられる way 数の上限数で変化させることができる.

### 4.3.2. 調査方法

調査方法として，Stubborn-Manually を用いて実行開始時点で Stubborn 戦略適用 way 数を 0 から 7 の値で固定して実行する.

### 4.3.3. 評価結果

調査の結果として，次の図 36，図 37 のグラフに LRU based Stubborn-Manually, DRRIP based Stubborn-Manually でのそれぞれのワークロード，使用 way 数ごとに，割り当て way 数 0 のときの IPC を標準としたときの IPC の増減をパーセントで示す. 凡例は Stubborn 領域として使用する way 数を意味する. 評価環境は 6 章で後述のものと同様である. これらのグラフを見るにあたって，way 数 4 のパターンが Normal Stubborn での評価結果に相当する.

この結果から，多くのワークロードにおいて Stubborn 戦略との親和性が高いワークロードでは Stubborn 戦略でセットを占める割合が高ければ高いほど性能への効果が高く，Stubborn 戦略と相性の悪いワークロードもまた Stubborn 戦略でセットを占める割合が高ければ高いほど性能低下が大きいという両極化が生じていることがわかる. また，性能が Stubborn 領域の割合に影響されず，ほとんど変化のないワークロードもある. これらの結果の中で，最も高い IPC を記録したときの way 数を表 3 に示す.

表 3 Stubborn-Manually 最大性能時 way 数

| | way | |
|---|---|---|
| | LRU | DRRIP |
| 410.bwaves | 2 | 2 |
| 429.mcf | 0 | 0 |
| 433.milc | 7 | 4 |
| 437.leslie3d | 0 | 0 |
| 450.soplex | 6 | 6 |
| 459.GemsFDTD | 0 | 0 |
| 462.libquantum | 1 | 0 |
| 470.lbm | 7 | 7 |
| 471.omnetpp | 7 | 7 |
| 473.astar | 3 | 1 |
| 482.sphinx3 | 6 | 7 |
| 483.xalancbmk | 7 | 7 |

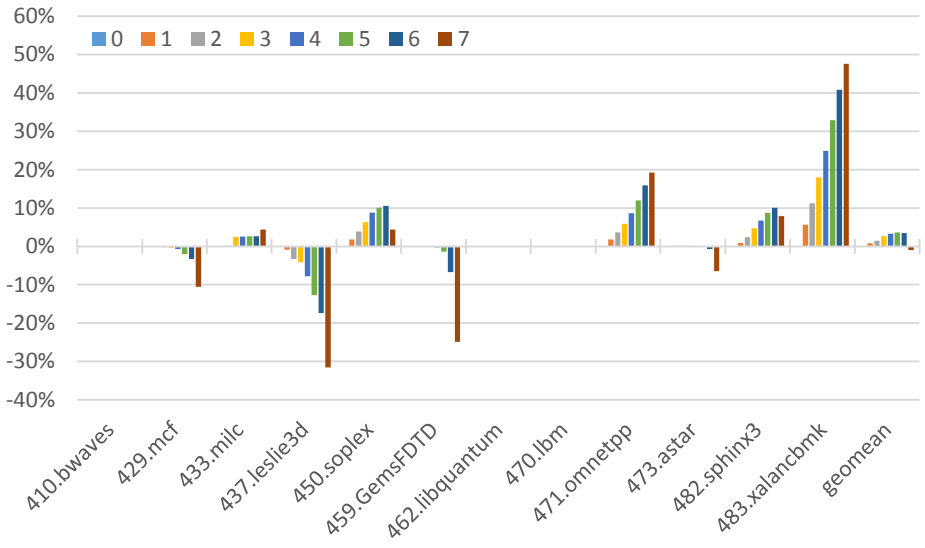



図 36 LRU based Stubborn-Manually での way 数別 IPC（0way を標準とした増減差分）

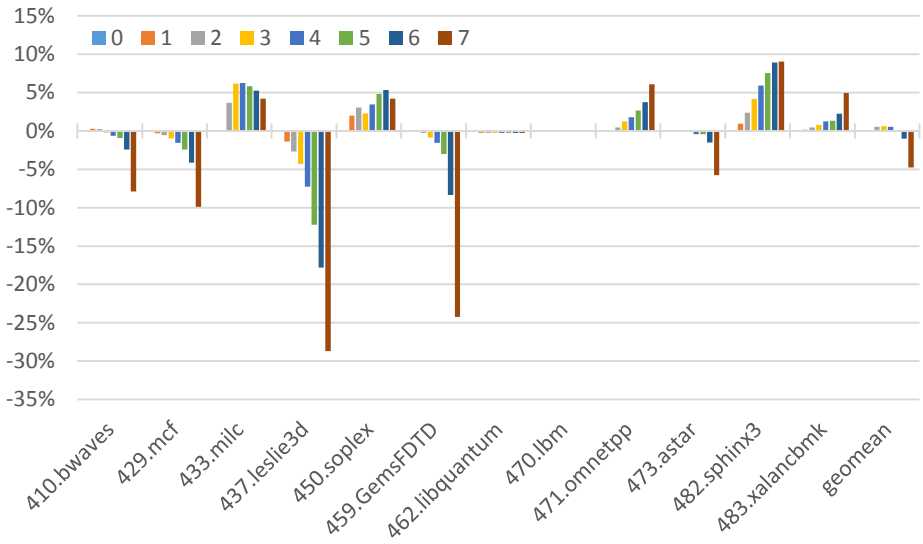



図 37 DRRIP based Stubborn-Manually での way 数別 IPC
（0way を標準とした増減差分）

### 4.3.4. まとめ

この調査のように静的に way 数を決めることで，表 3 に示す Stubborn-Manually 最大性能時 Stubborn 適用 way 数に近い結果が得られるワークロードもあれば，乖離するワークロードもある．この Stubborn 適用 way 数を実行時に変動させ，動的に理想の Stubborn 適用 way 数に近づけることができれば，Stubborn 戦略のより実用的な拡張となる．この結果を踏まえ，第 6 章では，Stubborn 領域の動的な変動による適応的な Stubborn 戦略の拡張を図る．

## 4.4. 全域 Stubborn キャッシュ

### 4.4.1. アイデア

4.3 節において，ワークロードの半数程度がより積極的な Stubborn 戦略の適用により性能を伸ばすことがわかった．そこで，単純な思考の延長として，LLC 全ての領域を Stubborn 領域として使用するとどうなるか，という疑問が生じる．一方で，先着順で定めたキャッシュラインだけを保持して，性能が維持できるか，という疑問もある．

しかしながら，このアイデアには魅力的な効果がある．それは，スラッシング耐性を考えるにおいて，OPT に相当する結果が得られる点である．3.2.4 節と同様にスラッシング耐性についてモデルを立てて考える．図 38 に示すこのモデルにおいて，Stubborn 領域を 100%とした全域 Stubborn キャッシュのポリシーのみが唯一 OPT の理論最適値と一致する．アクセスするアドレス帯が変動するなど，複雑な条件となるので Stubborn が常に OPT と同じ結果を示せるわけではないが，483.xalancbmk などのキャッシュアクセスに占めるスラッシングの割合の非常に高いワークロードにおける効果が期待できる．この疑問と期待から，LLC 全域を Stubborn 領域として使用する手法，Stubborn 100%について実装し，効果を調査した．

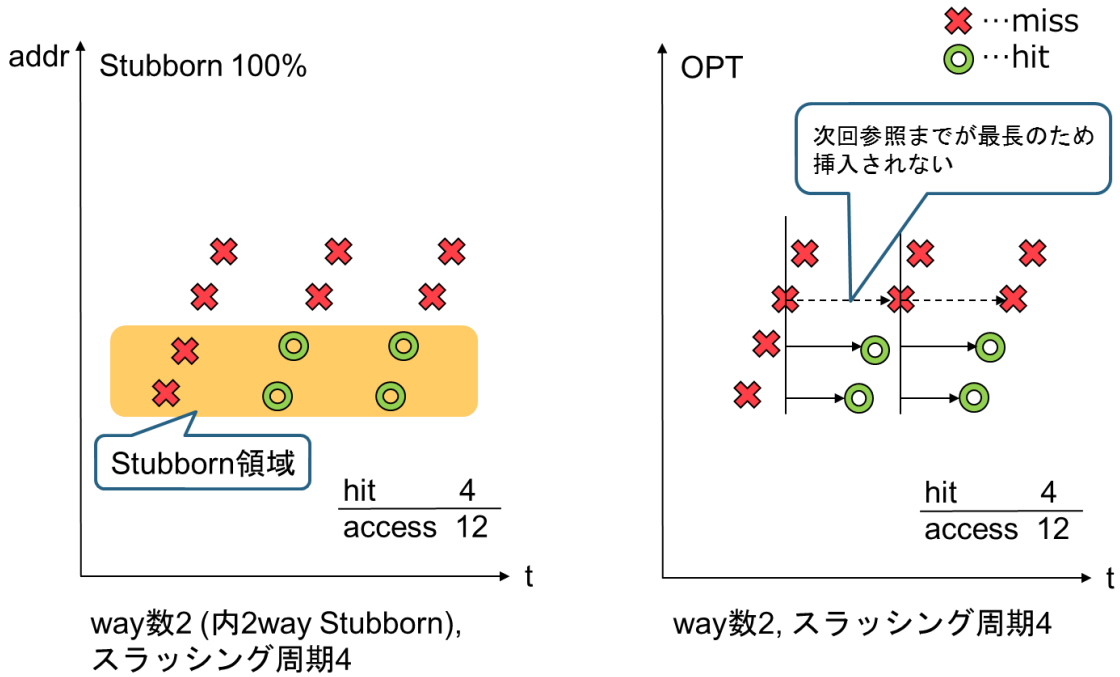



図 38 OPT と理論最適値と一致する Stubborn 100%のスラッシング耐性

### 4.4.2. 調査方法

Normal Stubborn の構成における Stubborn 戦略適用 way 数を 8, セットの全てのラインに Stubborn フラグを立てて置き換えを凍結できるものとした構成で実行する.

1 セット中 7 way までの Stubborn 戦略による凍結は，残りの 1 way によりデータの置き換えが可能となりメモリサブシステムでの上位の L2 キャッシュと下位のメインメモリとの接続機構に変更を加えずとも実現できる構成であったが，8 way 全てを凍結すると，メインメモリのデータへの参照で LLC へ新たなデータ挿入が必要となったタイミングでデータを渡すことができなくなりハングアップする．そこで，Stubborn 100%の実現にあたって，既に全てのラインで Stubborn フラグが立っているセットで生じたアクセスによるキャッシュミスについては LLC への新たなデータ挿入なしで上位階層（L2 キャッシュ）へ直接送るキャッシュバイパシングで対応する．メモリサブシステムとして，L2-メインメモリ間にバイパスを設ける.

評価では，LRU の性能を基準として LRU based Stubborn (Normal Stubborn)，DRRIP，DRRIP based Stubborn (Normal Stubborn)と比較する.

### 4.4.3. 評価

評価結果を次の図 39 に示す．LRU の IPC を 1 としたときの相対 IPC で比較する．結果より，Stubborn100%がこれまでの手法に勝るのは 483.xalancbmk 一点のみであり，そのベンチマークにおいても DRRIP ベースラインの Normal Stubborn の方がより高い性能を示している．プリフェッチ適性が高く Stubborn 戦略適用割合が大きいほど性能低下を起こすことがわかっている 429.mcf，437.leslie3d，459.GemsFDTD での結果はともかく，Normal Stubborn では高い性能向上を示していた 450.soplex，482.shpinx3 においても大きな性能低下が生じており，全ての way を凍結することでプリフェッチと短期で時間的局所性のある再参照を阻害することが大きなデメリットとなることが示された.

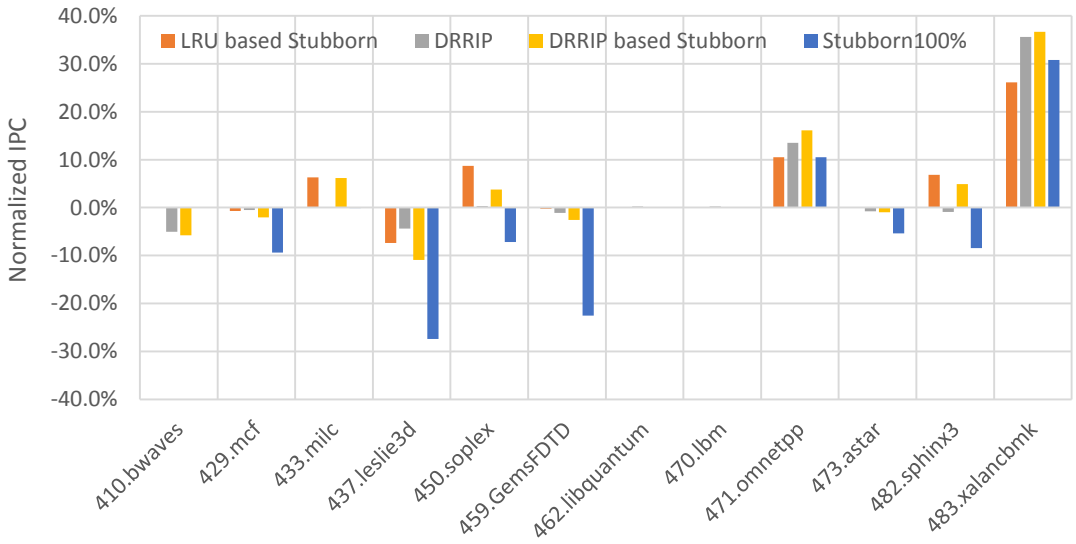



図 39 Stubborn100%の IPC 評価

## 4.5. まとめ

4 章では，Stubborn 戦略のもつ特性を複数のアプローチで解析し，調査によりポテンシャルを数字で示した．特に，4.2 節においては Stubborn 戦略を OFF にする期間を設けることで Normal Stubborn で生じていた副作用を抑える効果を確認し，また，4.3 節においては積極的な Stubborn 戦略の適用によりさらなる性能向上が得られることを確認した．

この結果を踏まえ，続く第 5 章では Stubborn 領域の動的な ON/OFF 切り替えにより Stubborn 戦略の副作用の抑制とさらなる性能向上を図る．第 6 章では，Stubborn 領域の動的な変動による適応的な Stubborn 戦略を提案する．

# 第5章 Stubborn 有効期間

## 5.1. コンセプト

第 4 章 4.2 節より，Stubborn 領域を常時有効とするナイーブな Normal Stubborn で生じる副作用に対して, フェーズに合わせた Stubborn 戦略の適用の ON/OFF 切り替えが有効であるということがわかった．これはつまり，Normal Stubborn における常時の Stubborn 戦略適用が副作用を産む原因であり，特にプリフェッチの恩恵が強く得られるワークロードでは大きな副作用をもたらしていたが，Stubborn 戦略を有効とするタイミングを制御することでこの副作用を抑えることができることを示している．しかしながら，フェーズの情報を用いた Stubborn 戦略の適用はこの調査のようにそれぞれのワークロードごとにフェーズの情報があらかじめ必要となるため，現実的ではない.

この考察を踏まえ，本章では Stubborn 戦略適用の判断を実行時に行い，動的に ON/OFF の切り替えを判断する手法を提案する．つまり，Stubborn 戦略を適用すべきタイミングでは適用し，Stubborn 戦略を適用すべきでないタイミングでは適用しない，という判断を実行時に為すための機構を提案する．ここで，この判断機構について 2 種類の構成を実装し，評価した.

## 5.2. SDM を用いた方法

ここまでの調査を踏まえ，Stubborn 戦略を適用することで効果が得られるタイミングと，Stubborn 戦略を適用すべきでないタイミングの判断を行う判断器を持つ置き換えアルゴリズムを提案する. ここで, 判断器として SDM を用いる手法を考案した. SDM におけるモニタリングセットで常時実行し比較するのは，ベースラインとする置き換えアルゴリズムと，ベースラインとする置き換えアルゴリズムに Stubborn 戦略を適用したものである．これを Stubborn-DYN と名づける．LRU をベースラインとする場合，SDM におけるモニタリングセットは 2 セットで, LRU を実行するセットと, LRU based Normal Stubborn を実行するセットである．この SDM に基づく Stubborn 戦略適用タイミングの判断を行う置き換えアルゴリズムを LRU based Stubborn-DYN と呼ぶ．DRRIP をベースラインとする場合，SDM におけるモニタリングセットは 3 セットで，LRU，SRRIP based Normal Stubborn，BRRIP based Normal Stubborn を実行するセットである．この SDM に基づく Stubborn 戦略適用タイミングの判断を行う置き換えアルゴリズムを DRRIP based Stubborn-DYN と呼ぶ.

### 5.2.1. 実装

LRU based Stubborn-DYN および DRRIP based Stubborn-DYN について，DRRIP からの差分である PSEL カウンタの扱いとフォロワーセットの動作について実装を記す.

#### 5.2.1.1. LRU based Stubborn-DYN の実装

実装として，LRU based Stubborn-DYN での PSEL カウンタの扱いについて記述する．LRU based Stubborn-DYN における SDM の構造を図 40 に示す．LRU を実行するモニタリングセットでのミスで PSEL カウンタを+1，LRU based Normal Stubborn を実行するモニタリングセットでのミスで PSEL カウンタを-1 する. フォロワーセットは，挿入時と追い出し時に PSEL カウンタを参照して動作を決定する.

PSEL カウンタが 0 以下のとき，フォロワーセットは LRU の動作を行う．追い出し動作では，LRU オーダーで最も LRU 側にあるラインを追い出し，挿入動作では新たなキャッシュラインを Stubborn フラグなしで挿入する.

PSEL カウンタが 0 より大きいとき，フォロワーセットは LRU based Normal Stubborn の動作を行う．追い出し動作では，Stubborn フラグが立っていないラインの中で，LRU オーダーで最も LRU 側にあるラインを追い出し，挿入動作では新たなキャッシュラインを，Stubborn フラグが上限数に達していなければ Stubborn フラグを立てて挿入する.

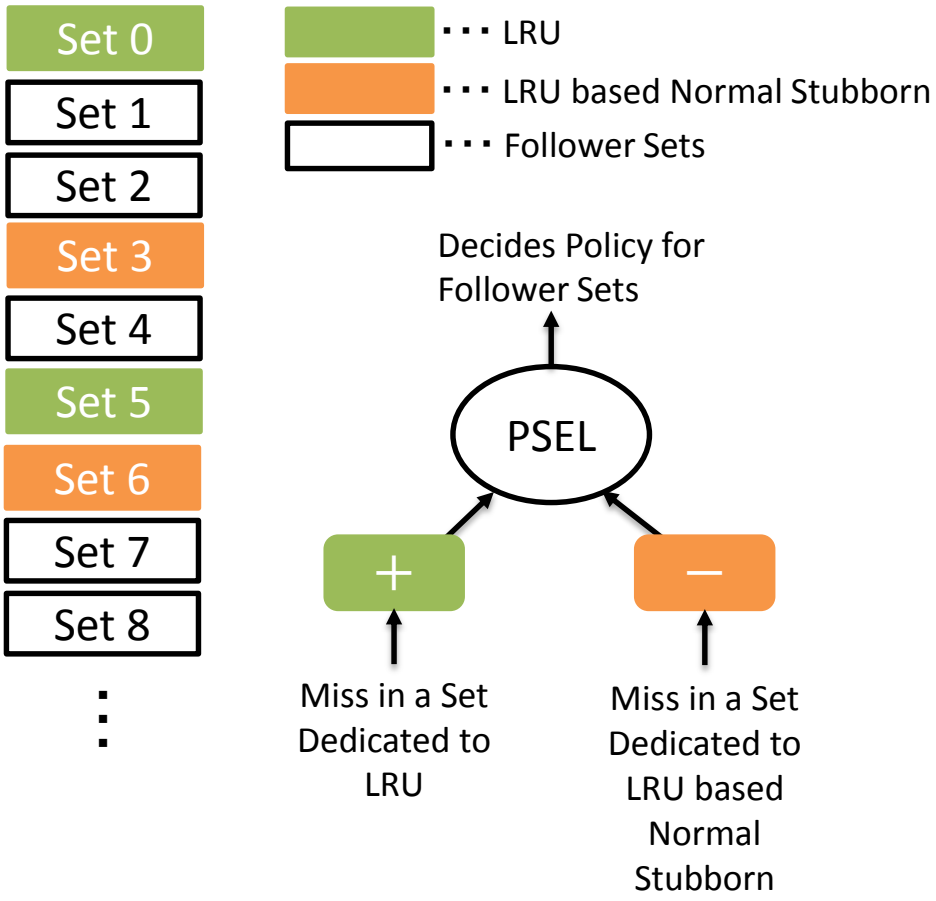



図 40 LRU based Stubborn-DYN における SDM

このような SDM を持つ LRU based Stubborn-DYN における，キャッシュテーブルへの参照時の動作を次の図 41 に図解する．この図では，キャッシュへの Read 要求が発生したとき，その要求データのアドレスについて，index が合致するセットの中に tag が照合するラインが存在せず，その結果キャッシュミスとなったときの動作を示している．キャッシュミスの判断により，より下層のメモリに問い合わせを送ると同時に，このケースではミスを生じたセットがモニタリングセットであったため，LRU based Stubborn-DYN における PSEL カウンタのルールに沿って，PSEL カウンタを-1 する．このカウント操作によって閾値である 0 を越えるか下回ることがあれば，そのタイミングでフォロワーセットにおけるポリシーが更新される．

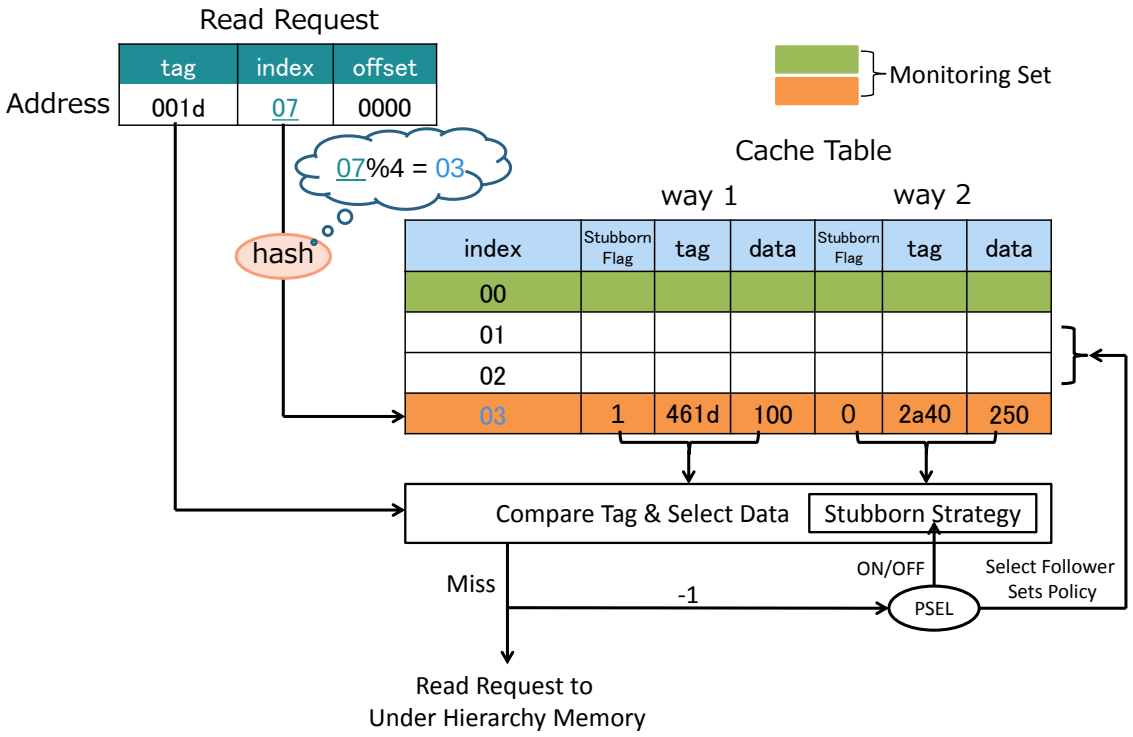



図 41 LRU based Stubborn-DYN における参照時の動作

また挿入時の追い出し判断についての動作を次の図 42 に図解する．この図では，index によって定まったセット内において，どのラインを追い出して新たなラインを挿入するかの判断をしている様子を示している．index によって定まった 03 のセット内には，Stubborn フラグが有効であるラインとフラグが立っていないラインがあり，またこの 03 のセットは LRU based Normal Stubborn のポリシーで動作するモニタリングセットであることから，LRU オーダーの条件（図では省略）と合わせて way 2 のセットが追い出し対象であることが定まる．このラインを追い出した後，新たに挿入するラインについて，現在のセット内の Stubborn フラグ数を確認して一定数以下であれば Stubborn フラグを立てて挿入する．

モニタリングセット以外への挿入時では，PSEL カウンタの値によってフォロワーセットが LRU ポリシーで動作するか LRU based Normal Stubborn のポリシーで動作するかを判断し，追い出し対象の選定において Stubborn フラグを勘案するかしないかを判断する．

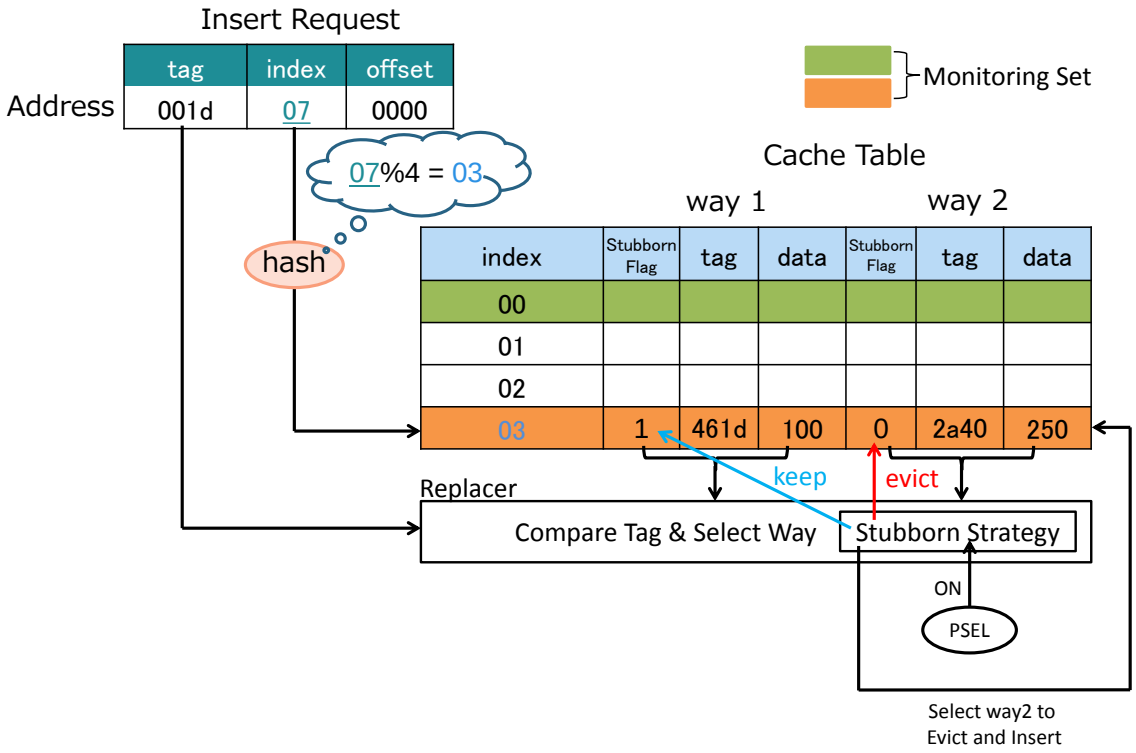



図 42 LRU based Stubborn-DYN における追い出し判断の動作

以降の章，節で述べる SDM と連携した Stubborn 戦略を用いる置き換えアルゴリズムの実装においても，この図 41，図 42 における参照と追い出し，挿入の動作をベースとして，その判断を行う PSEL カウンタの扱いや数，モニタリングセットとフォロワーセットに適用されるポリシー，SDM のフォロワーセットへの反映タイミングの違いなどによって実現される．そのため，差分がある部分についての解説を行い，参照時，挿入時の動作図については省略する．

### 5.2.1.2. DRRIP based Stubborn-DYN の実装

実装として，DRRIP based Stubborn-DYN での PSEL カウンタの扱いについて記述する．DRRIP based Stubborn-DYN における SDM の構造を図 43 に示す．LRU を実行するモニタリングセットでのミスで PSEL_LRU カウンタを+2，PSEL_SRST カウンタと PSEL_BRST カウンタを-1 する．SRRIP based Normal Stubborn を実行するモニタリングセットでのミスで PSEL_SRST カウンタを+2，PSEL_LRU カウンタと PSEL_BRST カウンタを-1 する．BRRIP based Normal Stubborn を実行するモニタリングセットでのミスで PSEL_BRST カウンタを+2，PSEL_LRU カウンタと PSEL_SRST カウンタを-1 する．フォロワーセットは，挿入時と追い出し時に PSEL カウンタを参照して動作を決定する．

フォロワーセットは，PSEL_LRU カウンタ，PSEL_SRST カウンタ，PSEL_BRST カウンタのうち最小の値を示すポリシーを選択し，その動作を行う．

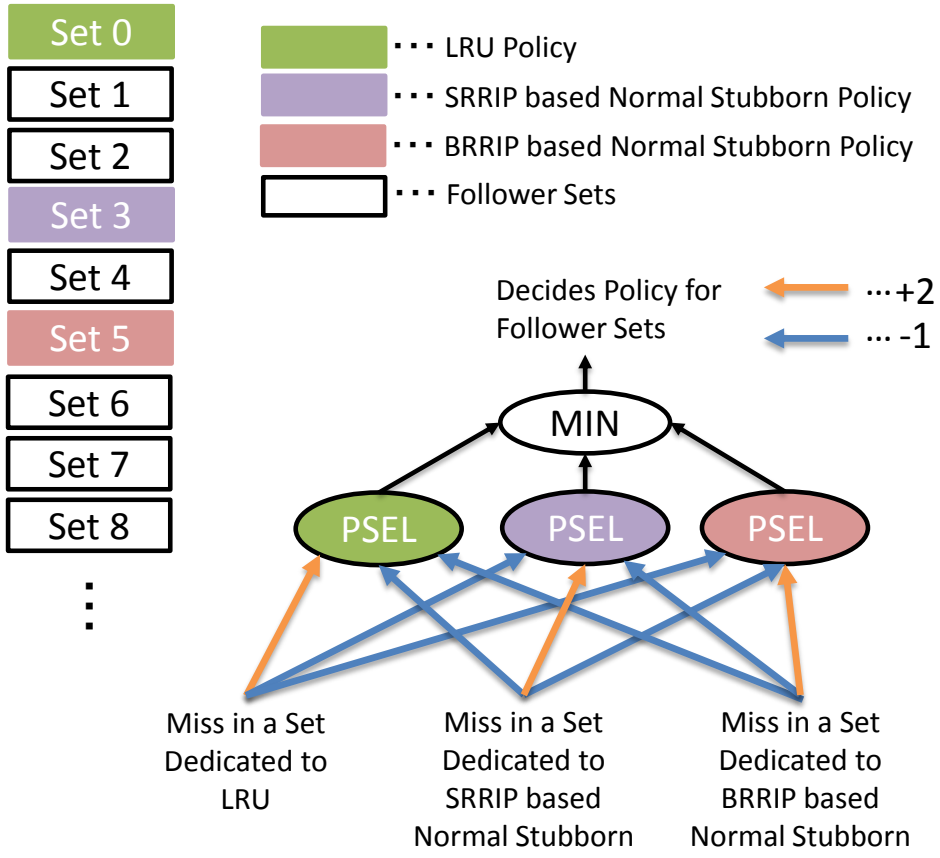



図 43 DRRIP based Stubborn-DYN における SDM

### 5.2.2. 評価

LRU based Stubborn-DYN の実行結果を図 44 に示す．性能評価として，ベースラインとする置き換えアルゴリズム（LRU）の IPC (Instruction per Cycle) を 1 としたときの相対 IPC の増減をパーセントで示す．437.leslie3d において，LRU based Normal Stubborn で生じていた副作用による性能低下を緩和している．また，483.sphinx3 では LRU based Normal Stubborn より高い性能を示した．しかしながら，LRU based Normal Stubborn で高い性能向上が得られていた 450.soplex では LRU 比でわずかに性能を落としており，LRU based Stubborn-DYN での判断が適切であるワークロードとそうでないワークロードがあることが示されている．

幾何平均では LRU に対して 0.6%の性能向上を示した. LRU based Normal Stubborn 比では幾何平均で 0.4%劣る結果となった．総評として，Stubborn 戦略による副作用を抑える効果はあるものの，性能向上も抑えられてしまう傾向にある．

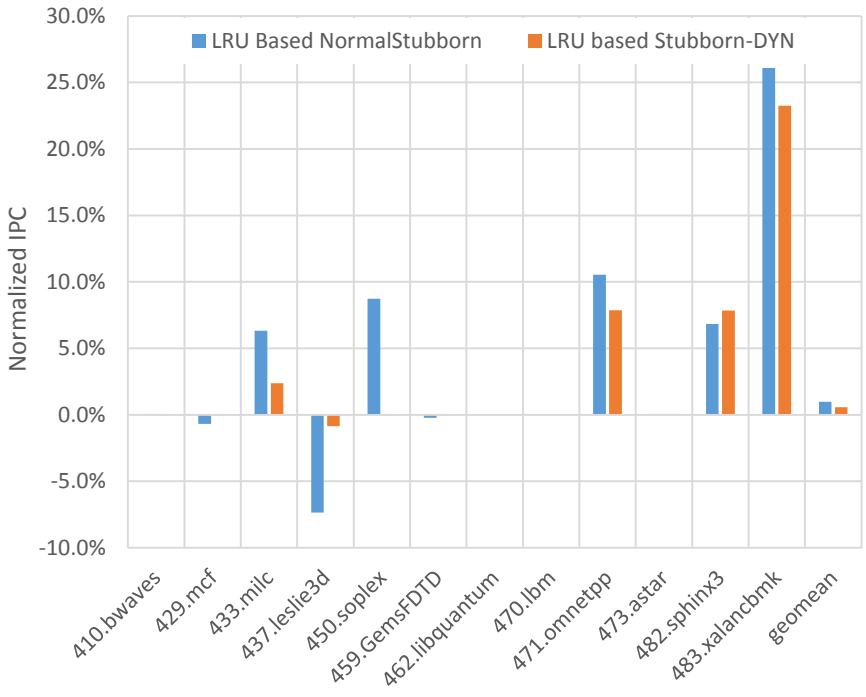



図 44 LRU ベースラインでの Stubborn-DYN 評価

次に，DRRIP based Stubborn-DYN の実行結果を図 45 に示す．性能評価として，ベースラインとする置き換えアルゴリズム（DRRIP）の IPC (Instruction per Cycle) を 1 としたときの相対 IPC の増減をパーセントで示す．

437.leslie3d において，DRRIP based Normal Stubborn で生じていた副作用による性能低下を緩和している．また，そのうえで 1.8%の性能向上が得られている．しかしながら，483.xalancbmk において DRRIP で得られていた性能向上が打ち消され，性能低下に転じており，DRRIP based Stubborn-DYN での判断が適切であるワークロードとそうでないワークロードがあることが示されている．

幾何平均では DRRIP に対して 0.7%の性能向上を示し，これは DRRIP based Normal Stubborn による DRRIP に対する 0.6%の性能向上を上回る結果となった．総評として，Stubborn 戦略による副作用を抑える効果があり，また副作用を抑えた上で性能向上に転じさせていることができている．

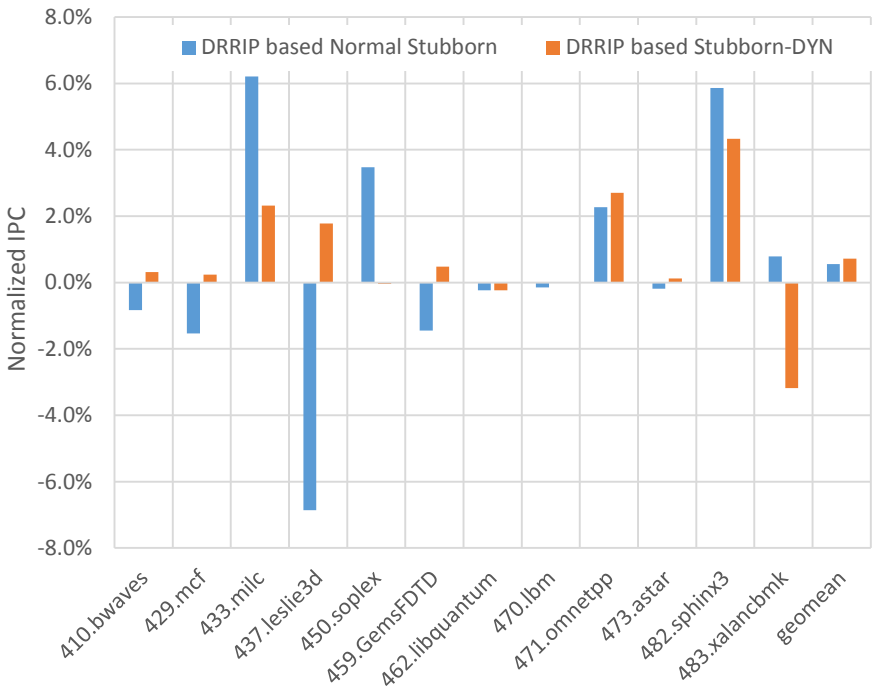



図 45 DRRIP ベースラインでの Stubborn-DYN 評価

## 5.3. BRRIP をトリガーとする方法

Stubborn 戦略はスラッシング耐性を持たせることを指向した置き換えアルゴリズムの戦略である．同じく，スラッシング耐性を持たせることを指向した置き換えアルゴリズムに BRRIP がある．これは 2.1.3 節でも解説したとおりだが，BRRIP には，DRRIP のなかで用いられる置き換えアルゴリズムとして，SDM の決定により適応的に用いられているという特徴がある．つまり，DRRIP の実行において，BRRIP はモニタリングセットの中でスラッシング耐性について適切に効果を発揮することができ，SDM はその結果を汲んで BRRIP をフォロワーセットに適用することができる．

そこで，BRRIP と Stubborn 戦略は親和性が高いのではないかと予想した．つまり，DRRIP の実行において BRRIP が適用されるタイミングは Stubborn 戦略を適用すべきタイミングでもあるのではないか，と推測できるということだ．そこで，Stubborn 戦略の適用タイミングの判断として BRRIP のフォロワーセットへの適用を利用した手法を発案した．この手法を BRRIP triggered Stubborn と呼ぶ．

BRRIP triggered Stubborn の Stubborn-DYN との違いは，SDM のモニタリングセットにおいて SRRIP，BRRIP 共に Stubborn 戦略を常時 OFF としている点である．

また，フォロワーセットのポリシーの BRRIP と SRRIP の切り替わりにおいて，Stubborn フラグを保持するかクリアするかによって 2 種類の手法を用意する．それぞれ BRRIP triggered Stubborn 1，BRRIP triggered Stubborn 2 と呼ぶ．

**BRRIP triggered Stubborn 1**

BRRIP から SRRIP に切り替わったときに Stubborn フラグは残すが target 時に無視する．SRRIP 期間では Stubborn フラグ付きのキャッシュラインも追い出される可能性がある．

**BRRIP triggered Stubborn 2**

BRRIP から SRRIP に切り替わったときに Stubborn フラグは全部クリアする．

BRRIP triggered Stubborn 1 では，短期間で BRRIP，SRRIP，BRRIP と切り替わった際に Stubborn フラグのついたキャッシュラインがセット内に残ることで，BRRIP に戻ったあとも継続して以前の BRRIP 実行時に保持していた Stubborn フラグ付きキャッシュラインを残すことができる．BRRIP triggered Stubborn 2 は，BRRIP triggered Stubborn 1 でのこの実装が効果を生じるかどうかを確認するための対照的な構成である．

### 5.3.1. 実装

BRRIP triggered Stubborn における SDM でのモニタリングセットのペアは，図 46 に示すように通常の DRRIP におけるモニタリングセットのペアと同じである．BRRIP triggered Stubborn では，この SDM による判断の結果，BRRIP が優勢であり，フォロワーセットに BRRIP が適用されている期間を Stubborn 戦略の適用期間として利用する．

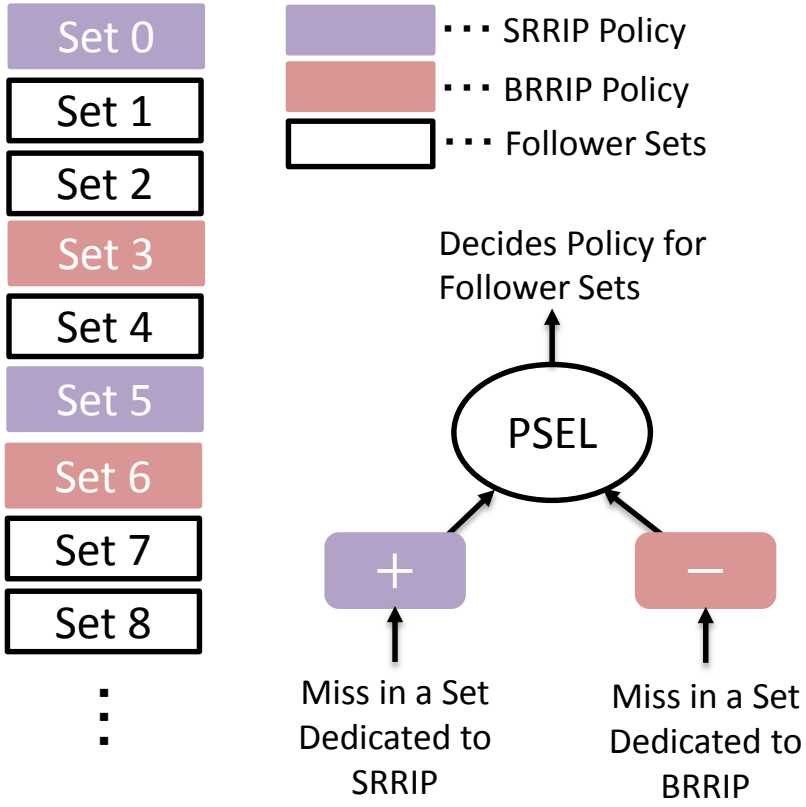



図 46 BRRIP triggered Stubborn における SDM

BRRIP triggered Stubborn 1 および BRRIP triggered Stubborn 2 について，DRRIP からの差分であるフォロワーセットの動作について実装を記す．

PSEL カウンタが 0 以下のとき，フォロワーセットは SRRIP の動作を行う．追い出し動作では，Stubborn フラグが立っていないラインの中で最も RRPV 値が高いラインを追い出し，挿入動作では新たなキャッシュラインを，Stubborn フラグが上限数に達していなければ Stubborn フラグ付きで挿入する．

PSEL カウンタが 0 より大きいとき，フォロワーセットは BRRIP based Normal

Stubborn の動作を行う．追い出し動作では，Stubborn フラグが立っていないラインの中で最も RRPV 値が高いラインを追い出し，挿入動作では新たなキャッシュラインを，Stubborn フラグが上限数に達していなければ Stubborn フラグ付きで挿入する．

## 5.3.2. 評価

BRRIP triggered Stubborn1,2 の実行結果を図 47 に示す．性能評価として，ベースラインとする置き換えアルゴリズム（DRRIP）の IPC（Instruction per Cycle）を 1 としたときの相対 IPC の増減をパーセントで示す．

BRRIP triggered Stubborn 1,2 共に，410.bwaves，429.mcf，437.leslie3d，459.GemsFDTD と，DRRIP based Normal Stubborn で副作用を生じていたワークロード全てにおいて，副作用による性能低下を緩和している．しかしながら，483.xalancbmk において大きく性能を落としており，BRRIP triggered Stubborn での判断が適切であるワークロードとそうでないワークロードがあることが示されている．

幾何平均では DRRIP に対して BRRIP triggered Stubborn 1,2 それぞれ 1.2%, 0.4% 劣る結果を示した．DRRIP based Normal Stubborn 比では 1.7%, 0.9%劣る結果となった．総評として，Stubborn 戦略による副作用を抑える効果はあるものの，別の副作用が大きく生じるケースがあった．

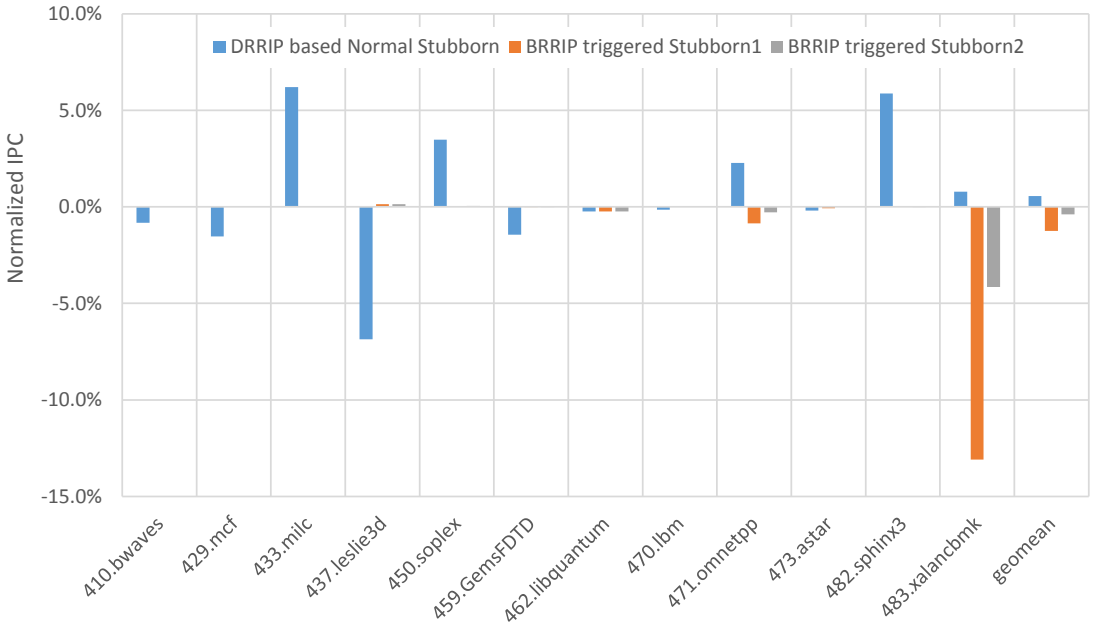



図 47 BRRIP triggered Stubborn の評価

## 5.4. まとめ

Stubborn 戦略を適用するタイミングの判断を行う判断器として SDM と，BRRIP の特性を利用した適応型の Stubborn キャッシュマネジメント手法を提案し，評価した．評価結果では，それぞれ適応的動作が適当となるワークロードと不適当なワークロードが存在するものの，副作用を抑制し，さらなる性能向上が得られるケースの存在を示した．

LRU based Stubborn-DYN では LRU に対して幾何平均で 0.6%の性能向上を示した．LRU based Normal Stubborn 比では幾何平均で 0.4%劣る結果となった．

DRRIP based Stubborn-DYN では DRRIP に対して幾何平均で 0.7%の性能向上を示し，これは DRRIP based Normal Stubborn による DRRIP に対する 0.6%の性能向上を上回る結果となった．

BRRIP triggered Stubborn 1,2 では DRRIP に対して幾何平均でそれぞれ 1.2%, 0.4%劣る結果を示した．DRRIP based Normal Stubborn 比では 1.7%, 0.9%劣る結果となった．

# 第6章 Stubborn 領域の増減

## 6.1. コンセプト

第 4 章 4.3 節より， Stubborn 領域と従来手法を半分半分で動かすというナイーブな Normal Stubborn 戦略の適用に対して，Stubborn 戦略との適応性が高いワークロードでは Stubborn 領域をより多く取るべきであり，また Stubborn 戦略と相性の悪いワークロードでは Stubborn 領域は与えないでいるべきだということがわかった．これはつまり，Normal Stubborn における半分で固定された Stubborn 領域の割合は中庸な選択であり中庸な結果をもたらしていたということを意味するが，Stubborn 領域として用いる way 数を適応的に最小または最大のどちらかから選ぶ制御をすることでより高い性能を得ることができると予想される．

この考察を踏まえ，本章では Stubborn 領域を動的に変動させる手法を提案する．つまり，Stubborn 領域を大きくとるべきワークロードでは大きくとり，Stubborn 領域を取るべきではないワークロードでは取らない，という判断を実行時に為すための機構を提案する．ここで，この判断機構を持つ Stubborn 戦略の実装をトランプゲームの『High and Low』になぞらえて Stubborn-HL (High & Low)と名付ける．

## 6.2. High and Low

ここで，Stubborn 領域を大きくするか小さくするかを判断するための機構として SDM を使用する．SDM で用いるモニタリングセットの片方に Stubborn 領域を大きく取るポリシーを，もう片側に小さく取るポリシーを適用し，成績の良かった側がその他全体のセットに適用される，という動作をする．

LRU をベースとする Stubborn-HL（以下 LRU based Stubborn-HL）においては大きくするか小さくするかの 2 方向の判断をするだけで良いのでモニタリングセットは 2 セット， PSEL カウンタも 1 つで実装可能となる．一方，DRRIP をベースとする Stubborn-HL（以下 DRRIP based Stubborn-HL）では，DRRIP 内で SRRIP と BRRIP を切り替えるために既に SDM が用いられている．ここに Stubborn-HL を適用するためには，さらに追加のモニタリングセットが必要となる．

ここで，原著[22]における本来の SDM は直近のアクセス傾向を重視して， PSEL カウンタが更新される度に随時全体に反映する手法を選択・適用するが，この挙動は一定以上の長期再参照によるミスを防ぐことを目的としている Stubborn 戦略の効果を打ち

消してしまうことを，第 5 章における SDM を判断器とする置き換えアルゴリズムの動作から知見として得た．その理由は，Stubborn 戦略が効果的に働くのは長期の再参照を保護するケースであるのに対して，PSEL カウンタの値による随時の判断では，最短で数命令ごとに Stubborn 戦略の適用を ON/OFF させてしまい，その度に長期に温存されているべきキャッシュラインを追い出す判断がなされてしまうためである．

そのため，本研究では PSEL カウンタの値による SDM の切り替え判断を随時ではなく一定以上の間隔をもって行うように修正して適用する．

また SDM による Stubborn 領域を適用する way 数の変動のさせ方について，第 4 章 4.3 節の結果から，本研究では way 数を細かく変動させるよりも大きい値と小さい値のどちらかに切り替える手法を選択した．本研究では Stubborn 領域を全く適用しない 0 way と Stubborn 領域を可能な限り使用する 7 way の両極な 2 種類をモニタリングセットで動作させ，これらのうちモニタリング期間で優れた成績を出した方をその他のセットのポリシーとして適用させる．つまり，0 way として指定されたセットではベースとなる置き換えアルゴリズムがそのまま動作し，7 way としたセットでは残りの 1 way にのみ追い出しが発生する．このような設計をもって SDM を用いることで，本来短期的な決定を下す SDM を Stubborn 戦略のような長期的な判断が必要な機構と組み合わせることができるようになる．

### 6.2.1. 実装

#### 6.2.1.1. Stubborn フラグビット

これまでの Stubborn 戦略と同様に，キャッシュテーブル中の各キャッシュラインに Stubborn フラグとして 1bit のフラグを追加する．挿入時の Stubborn フラグを立てるか立てないかの判断は，Normal Stubborn 戦略と同様にセットあたりの Stubborn フラグビットの許容数を満たすまでの間先着順で ON にする方法とする．

### 6.2.1.2. SDM の実装

LRU based Stubborn-HL において，10bit の PSEL カウンタを 1 つ，モニタリングセットを 64 セットあたり 2 セット設け，1 つでは Stubborn フラグビットのセット数上限を 7 way とした Stubborn 戦略動作，もう 1 つでは 0 way，つまり Stubborn 戦略に基づく動作をしない通常の LRU 動作を行う．それぞれのモニタリングセットでキャッシュミスが発生した際，0 way 側では PSEL カウンタを+1，7 way 側では-1 する．構造を図 48 に示す．

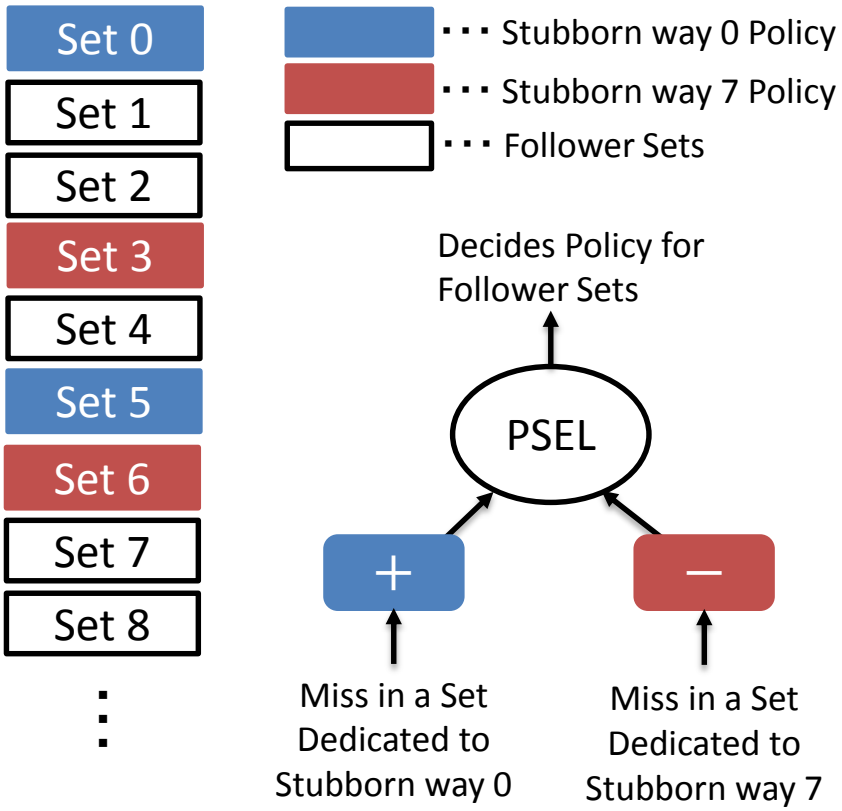



図 48 LRU based Stubborn-HL における SDM

DRRIP based Stubborn-HL において 10bit の PSEL カウンタを 4 つ，モニタリングセットを 64 セットあたり 4 セット設け，順に SRRIP + Stubborn 0 way，SRRIP + Stubborn 7 way，BRRIP + Stubborn 0 way，BRRIP + Stubborn 7 way の 4 種の置き換えアルゴリズムで動作する．ここで，SRRIP + Stubborn 7 way と BRRIP + Stubborn 7 way は，残りの 1way での SRRIP・BRRIP 動作をすることになり，長期的に見ると結果としてほぼ同じ動作を行うことになるが，これは Stubborn 7 way のモニタリングセットを 1 組だけにした場合，0 way のモニタリングセット 2 組との PSEL カウンタの増減のやりとりになり不利になってしまうため，平等性のために 0 way になるモニタリングセットと 7 way になるモニタリングセットの数を合わせるためにこのような実装となっている．より具体的には SRRIP + Stubborn 7 way と BRRIP + Stubborn 7 way のセットについては共に LRU + Stubborn 7 way とする単純化が行える．

それぞれのモニタリングセットでキャッシュミスが発生した際，該当のモニタリングセットの PSEL カウンタを+3，他の PSEL カウンタを-1 する．構造を図 49 に示す．

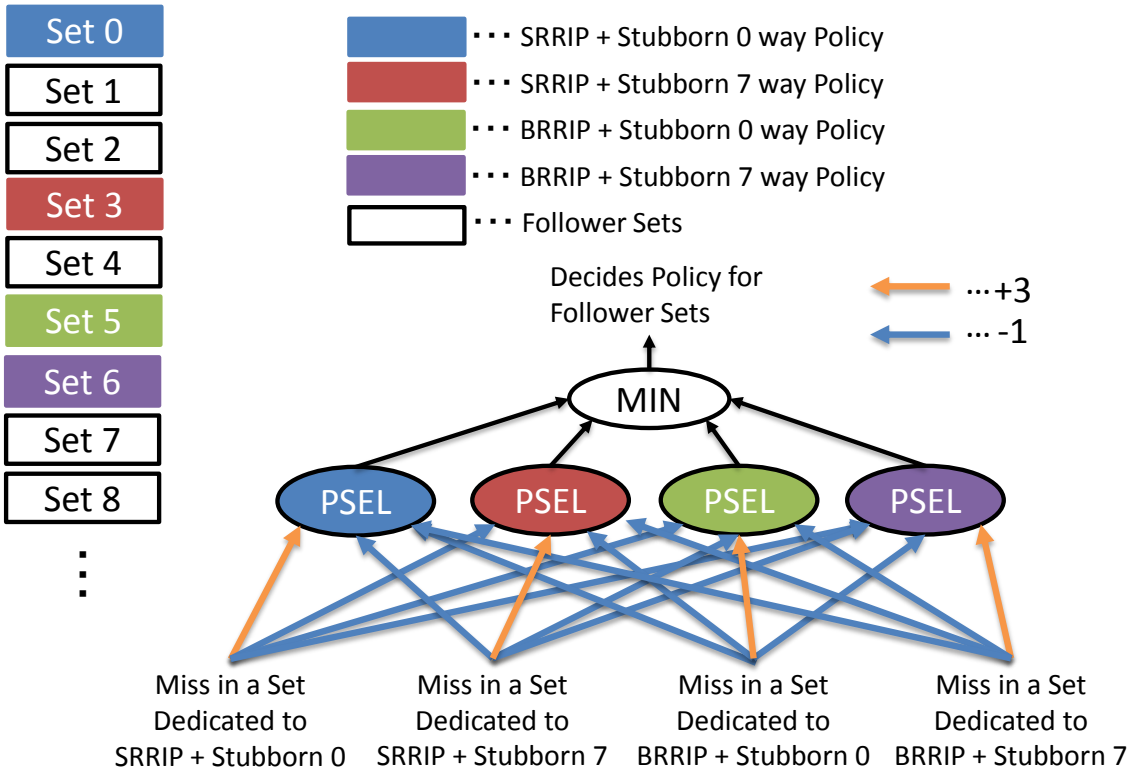



図 49 DRRIP based Stubborn-HL における SDM

#### 6.2.1.3. SDM 決定間隔

Stubborn-HL では，SDM による支配的なポリシーの選択を一定以上の間隔を空けて行う．この時間間隔を SDM 決定間隔と呼ぶ．実装においてはこの間隔を，実行命令数（経過命令数）を用いて指定する．このため，命令実行数を記録できるカウンタをもつ．いずれかのキャッシュラインでミスが生じた際にカウンタの値が決定間隔として指定された命令数を越えていれば，そのタイミングで最も成績の良いポリシー，LRU based Stubborn-HL では PSEL 値がマイナスまたは 0 であれば 7 way，プラスであれば 0 way，DRRIP based Stubborn-HL であれば PSEL 値の最も低いポリシーをモニタリングセット以外の全てのキャッシュセットのポリシーとしてセットし，次の SDM 決定間隔が経過するまで保持する．また，実行開始から初回の SDM 決定間隔の経過までの間，Stubborn way 数は間をとって 4 とする．

#### 6.2.1.4. PSEL リセット

決定間隔を長く持たせた SDM において，いずれかのポリシーに強く振れるタイミングがあった後，その後の支配的なポリシー判断においてその局所的なアクセス履歴による影響が強く長く残ってしまう場合がある．これを防ぐために，SDM 決定間隔が経過してポリシーの全体反映をする度に PSEL カウンタの値を全て 0 リセットする手法と，前回までの影響も残しつつ緩和するために PSEL カウンタの値を半分にする手法を考案し実装した．これらは手法名としてそれぞれ Stubborn-HL-Reset, Stubborn-HL-Half といったようにサフィックスの形で表現する．

## 6.2.2. 評価

### 6.2.2.1. 評価環境

置き換えアルゴリズムについて，LLC にベースラインとして Stubborn 戦略を使用しない No Stubborn (LRU, DRRIP)，ナイーブな手法としての Normal Stubborn，提案手法として Stubborn-HL，Stubborn-HL-Half，Stubborn-HL-Reset の 3 種を合わせた 5 種類の置き換えアルゴリズムを適用して実行する．

評価の節では，これに参考値として Stubborn-Manually でのワークロードごとのベスト値で揃えた結果（以降，Stubborn-Manually-Best）を加えた計 6 種類での結果を，LRU をベースとしたもの，DRRIP をベースとしたものそれぞれで比較評価する．
パラメータとして，SDM で用いる PSEL カウンタは 10bit，DRRIP で用いる RRPV は 2bit，SDM 決定間隔は 20M 命令とする．これらの値は予備評価を行って決定した．

### 6.2.2.2. 評価結果

#### 6.2.2.2.1. IPC

性能評価として，ベースラインとする置き換えアルゴリズム（LRU, DRRIP）の IPC (Instruction per Cycle) を 1 としたときの相対 IPC の増減をパーセントで示す．LRU をベースラインとしたものを図 50, DRRIP をベースラインとしたものを図 51 に示す．

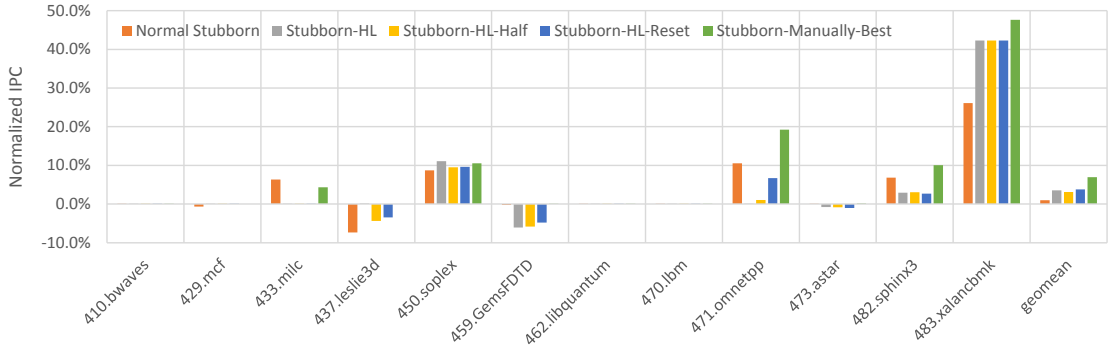



図 50 LRU ベースラインにおける相対 IPC による Stubborn-HL 評価

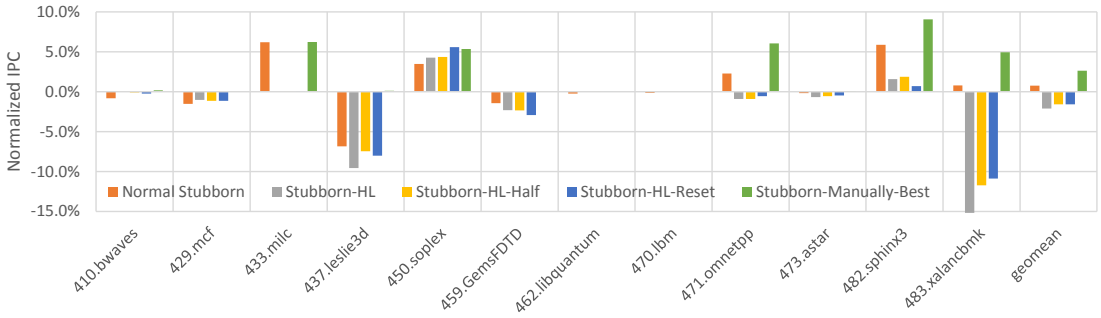



図 51 DRRIP ベースラインにおける相対 IPC による Stubborn-HL 評価

LRU ベースラインの評価（図 50）において，提案手法 Stubborn-HL-Reset により LRU 比で最大 42.3%，幾何平均で 3.8%の性能向上が得られている．Normal Stubborn 比では 2.8%の性能向上が得られた．

Stubborn-Manually-Best に対しても，ワークロード 450.soplex において Stubborn-HL が 0.5%勝る．437.leslie3d でもわずかながら 0.1%の向上が得られ，同時にこのケースでは Normal Stubborn で生じていたベースラインからの 7.4%の性能低下を抑えた上で得られた成績である．

一方，DRRIP ベースラインの評価（図 51）にて，Stubborn-HL-Reset では DRRIP 比で最大 5.6%の性能向上が得られたものの，幾何平均では 1.6%の性能低下が生じた．

LRU ベースラインの提案手法と DRRIP ベースラインの提案手法の比較では，LRU ベースラインで最善の LRU based Stubborn-HL-Reset が DRRIP ベースラインで最善の DRRIP based Stubborn-HL-Reset に対して 2.5%高く， DRRIP に対して 1.7%，Normal Stubborn DRRIP に対しても 0.9%の差で高い性能を示した．

### 6.2.2.3. MPKI

性能評価に付随して，キャッシュミス数を MPKI で示す．LRU をベースラインとしたものを図 52, DRRIP をベースラインとしたものを図 53 に示す. ミスの削減率が IPC の向上率と概ね対応しており，結果を裏付ける数値が出ている．

410.bwaves，462.libquantum，470.lbm において MPKI の値が極端に低いのは，これらのワークロードではプリフェッチが極めて有効なためである. そして，この結果は，提案手法による Stubborn 戦略適用時にもプリフェッチを阻害しなかったことも意味している．

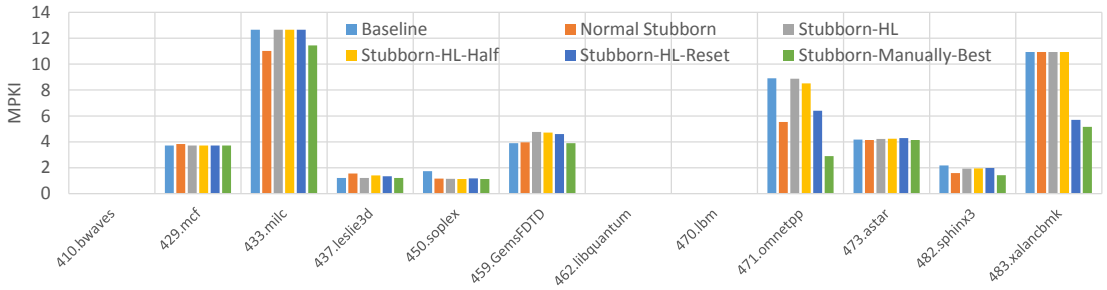



図 52 LRU ベースラインにおける MPKI による Stubborn-HL 評価

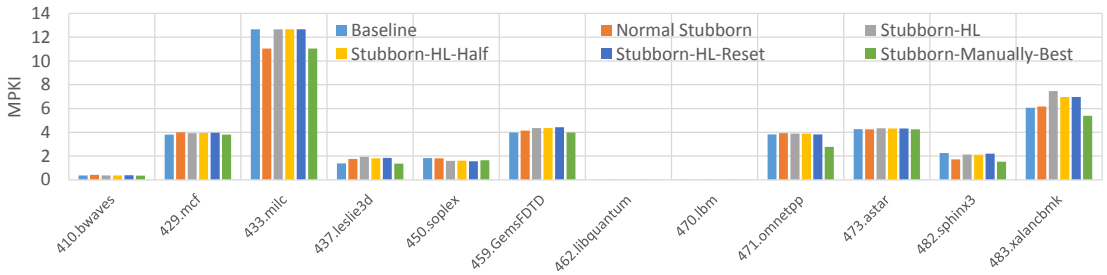



図 53 DRRIP ベースラインにおける MPKI による Stubborn-HL 評価

## 6.3. 考察

### 6.3.1. SDM の判断

SDM による Stubborn 領域の割り当ての判断の評価として，それぞれの提案手法による way 数の判断がどの程度 Stubborn-Manually-Best の判断に漸近しているかを確認した．ここで，評価のため，一定時間間隔で way 0,7 で切り替えられた way 数の判断の Stubborn-Manually-Best に対する一致率を以下の手順により集計した．

1. シミュレーション実行時，SDM 決定間隔で定めた命令数経過ごとに
   採択 way 数（0 or 7）を記録する
2. この記録を集計し，提案手法・実行ワークロードごとにその平均値を求める
3. 2 で求めた平均値から Stubborn-Manually-Best での way 数を引く
4. 3 で求めた値の絶対値をとり，これを Stubborn-Manually-Best による
   判断 way 数からの差とする
5. 4 で求めた差を最大 Stubborn way 数の 7 で割り，
   これを Manually-Best からの乖離率とする
6. 1 から乖離率を引いて一致率とする

手順 2 の時点での各ワークロード・手法ごとの Stubborn-Manually-Best での各 way 数と提案手法での Stubborn 領域使用 way 数の平均値を LRU ベースラインのときのものを表 4，DRRIP ベースラインのときのものを表 5 に示す．この表から，求められている way 数に対する，提案手法によって判断した way 数の平均値の過不足が読み取れる．

また，手順 6 で求めた一致率を LRU ベースラインのときのものを図 54，DRRIP ベースラインのときのものを図 55 に示す．IPC, MPKI と合わせて見ると，Stubborn-Manually-Best での way 数の一致率が高いほどミスが少なく IPC が高い傾向にある．個々の例を見ていくと，LRU ベースラインでは，提案手法が Stubborn-Manually-Best に並んで高い成績を出した 450.soplex では，LRU ベースライン，DRRIP ベースライン共に高い way 数再現性と共に，高い IPC の再現も得られている．482.sphinx3 についても同様の傾向が見られる．

483.xalancbmk においては，全ての提案手法が全ての判断タイミングで way 7 の判断を下せており，Stubborn-Manually-Best の実行と高く一致する．Stubborn-Manually-Best と提案手法の間に残る性能差は初回の SDM 決定間隔までの間の挙動の

差と，常時 0 way で稼働するモニタリングセットの片側で生じるロスによるものである．一方で，DRRIP ベースラインでは 6.4～6.6 と，完全ではないものの高い再現性は得られていたにもかかわらず，IPC では 100%を下回る結果となった．これは，実行時に 0 way にする判断が 7 にする判断に混じったことで，実行の当初から保持されているべきキャッシュセットで 0 way の判断が降りる都度に追い出されてしまい，483.xalancbmk で生じる激しいスラッシングへの耐性を失ってしまったためである．このことから，483.xalancbmk においては LRU ベースラインでの提案手法の判断のように，1G 命令期間中，Stubborn 領域に留めるキャッシュラインを一切更新しないことが有効であるとわかった．この結果は，483.xalancbmk のような過密で長いスラッシングの連続するワークロードにおいては，キャッシュラインを入れ替えないことだけが唯一置き換えアルゴリズムにできる対処であることも示している．

LRU ベースラインでの 437.leslie3d の実行では，Normal Stubborn で生じている性能低下を，提案手法では Stubborn 適用割合を動的に抑えて，性能低下も小さく抑え，克服することができている．このケースでは，プリフェッチとの親和性が高い 437.leslie3d のようなワークロードで，Stubborn 適用 way 数を 0 としたことでプリフェッチを阻害せず，ベースラインで得られていた性能を維持することができている．

このように，Stubborn-Manually-Best で低い Stubborn 適用割合を示すワークロードにおいて，提案手法により Stubborn 適用割合を低く抑えられたことでスラッシング耐性が得られ，プリフェッチとの親和性が得られた．

LRU ベースラインと DRRIP ベースラインでの比較では，LRU ベースライン（図 54）では Stubborn-Manually-Best の一致率を 100%としたとき，一致率は LRU に対して Normal Stubborn が 14.3%，Stubborn-HL-Reset が 17.2%の向上を果たした．DRRIP ベースライン（図 55）では DRRIP に対して Normal Stubborn が 7.1%，Stubborn-HL-Reset が 5.1%の向上となった．結果として，DRRIP ベースラインでの Stubborn 戦略適用より，LRU ベースラインでの Stubborn 戦略適用の方が SDM による判断がより良くなっている．これは，IPC での順列とも合致する．

表 4 平均 Stubborn way 数（LRU ベースライン）

| Workload | Manually | HL | HL-Half | HL-Reset |
|---|---|---|---|---|
| 410.bwaves | 2.0 | 0.0 | 0.1 | 0.7 |
| 429.mcf | 0.0 | 0.0 | 0.0 | 0.0 |
| 433.milc | 7.0 | 0.0 | 0.0 | 0.6 |
| 437.leslie3d | 0.0 | 0.1 | 2.7 | 2.4 |
| 450.soplex | 6.0 | 7.0 | 5.7 | 6.0 |
| 459.GemsFDTD | 0.0 | 7.0 | 5.4 | 4.4 |
| 462.libquantum | 1.0 | 7.0 | 5.4 | 5.3 |
| 470.lbm | 7.0 | 6.7 | 5.4 | 4.3 |
| 471.omnetpp | 7.0 | 0.0 | 0.6 | 3.4 |
| 473.astar | 3.0 | 7.0 | 6.7 | 5.3 |
| 482.sphinx3 | 6.0 | 1.9 | 2.0 | 1.9 |
| 483.xalancbmk | 7.0 | 7.0 | 7.0 | 7.0 |

表 5 平均 Stubborn way 数（DRRIP ベースライン）

| Workload | Manually | HL | HL-Half | HL-Reset |
|---|---|---|---|---|
| 410.bwaves | 2.0 | 2.3 | 3.3 | 2.3 |
| 429.mcf | 0.0 | 2.0 | 2.1 | 2.1 |
| 433.milc | 4.0 | 1.0 | 0.9 | 0.7 |
| 437.leslie3d | 0.0 | 3.6 | 3.1 | 3.4 |
| 450.soplex | 6.0 | 5.3 | 4.7 | 5.3 |
| 459.GemsFDTD | 0.0 | 2.7 | 3.0 | 3.7 |
| 462.libquantum | 0.0 | 4.7 | 4.6 | 5.1 |
| 470.lbm | 7.0 | 3.9 | 3.9 | 3.4 |
| 471.omnetpp | 7.0 | 1.9 | 1.9 | 1.6 |
| 473.astar | 1.0 | 3.6 | 3.7 | 3.4 |
| 482.sphinx3 | 7.0 | 1.3 | 1.6 | 0.9 |
| 483.xalancbmk | 7.0 | 6.4 | 6.7 | 6.6 |

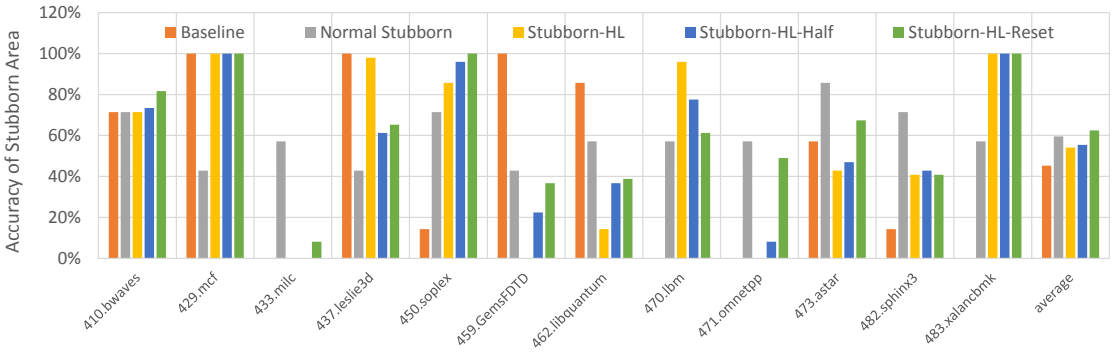



図 54 LRU ベースラインでの Stubborn 領域割合判断の一致率

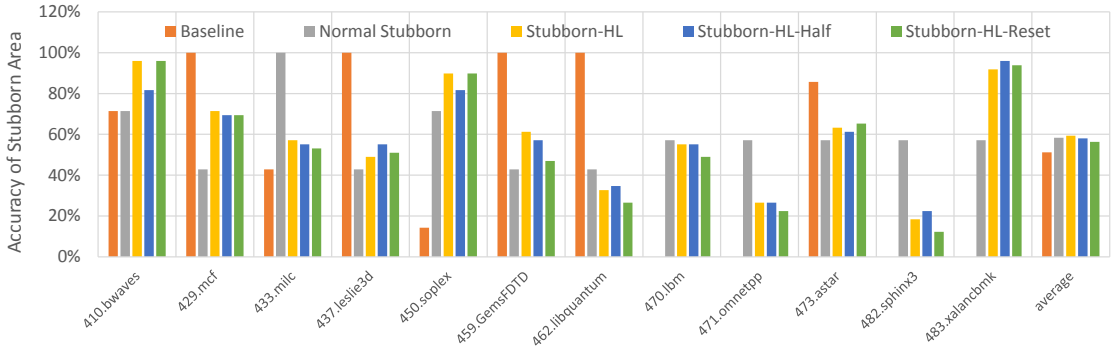



図 55 DRRIP ベースラインでの Stubborn 領域割合判断の一致率

### 6.3.2. ハードウェアコスト

ベースライン, Normal Stubborn, 提案手法におけるハードウェアコストを評価する. 表 6 に予測機構とキャッシュテーブルで用いるハードウェア量, その容量に対するオーバーヘッドを示す. ここで, 予測機構とは SDM のための PSEL カウンタを意味し, キャッシュテーブルにおけるハードウェア量は LRU では LRU オーダー, DRRIP では RRPV, Stubborn 戦略ハイブリッドのポリシーではそれらに Stubborn フラグを加えた置き換えアルゴリズム状態管理のための記憶領域を指す. これより, LRU ベースライン, DRRIP ベースライン共に, 提案手法は 4KB のテーブルサイズ増加, 2MB のキャッシュ容量に対して 0.2%のハードウェア増加コストで実現できる. このコストは得られた性能向上に対して十分に小さい.

表 6 ハードウェアコストの比較 (8 way 2MB LLC)

| Policy | | Predictor Structures | Cache Tag Meta-data | Overhead for Capacity |
|---|---|---|---|---|
| LRU based | LRU | None | 12KB | 0.6% |
| | Normal Stubborn | None | 16KB | 0.8% |
| | Stubborn-HL | 10 bit | 16KB | 0.8% |
| DRRIP based | DRRIP | 10 bit | 8KB | 0.4% |
| | Normal Stubborn | 10 bit | 12KB | 0.6% |
| | Stubborn-HL | 40 bit | 12KB | 0.6% |

## 6.4. まとめ

Stubborn 戦略の活用タイミングを適応的に決定する手法を提案し，これにより性能向上を維持したまま，性能低下の発生を抑えることを両立する機構を実現した．シミュレータによる評価では，LRU に対して最大 42.3%，幾何平均で最大 3.8%の性能向上を示した．Normal Stubborn と比較した場合でも 2.8%の性能向上となり，特に，7.4%性能低下していたワークロードでは性能低下を抑えて 0.1%の性能向上が得られるまで改善が見られるなど，提案の適応制御により性能を安定させる効果が確認された．

プリフェッチについても，本章でのストリーミングプリフェッチとの共存の結果が示したように，提案手法により Stubborn 領域の判断が正しくなされることで，プリフェッチが優先して積極的に動くべきタイミングで Stubborn は無効（Stubborn 領域 0 way）になり，プリフェッチを阻害することなく共存することができる．

IPC，MPKI，SDM による判断，ハードウェアコストを総合した結論としては，適応型の Stubborn 戦略を組み合わせた置き換えアルゴリズムとして LRU ベースラインであり更新時 PSEL リセットの構成による提案手法，LRU based Stubborn-HL-Reset が第 6 章における最大の成果物である．

# 第7章 結論

## 7.1. 本研究の成果

本研究では，まず LLC で生じるキャッシュミスについての調査を行い，LLC で生じるキャッシュミスの多くを占めるものが再参照ミスであり，また従来手法では対処できない，残されたキャッシュミスが 10～100M（メガ）の長期の命令数間隔に渡る再参照であることを明らかにした．

この調査結果を踏まえ，直近の履歴やその学習によって得られるキャッシュラインの保持の選択とは異なるアプローチを採る Stubborn 戦略を提案した．これは，キャッシュラインの追い出しを一時的に凍結し，キャッシュラインの入れ替えを起こさない領域を設けることで，長期保持による統計的なヒット数向上を可能とするような戦略である．この Stubborn 戦略の最も単純な実装である Normal Stubborn では，従来の手法では的確な追い出し選択が困難であった 10～100M 命令間隔の長期再参照が連続するワークロードや，激しいスラッシングによりキャッシュ容量が全く活用されないワークロードなどに有効であることを確認した．さらに，ワークロードのメモリアクセス傾向を詳細に調べ，モデルと共に例示することで Stubborn 戦略が有効であったワークロードにおいて，キャッシュラインの凍結という一見非効率な Stubborn 戦略が何故 LLC において有効に機能するか明らかにした．

Normal Stubborn をベースとして，Stubborn 戦略が持つ特性を調査することにより Stubborn 戦略の適用領域の増減によりさらなる性能向上が得られることを明らかにした．さらに，判断器として SDM を用いて Stubborn 戦略の適用タイミングを適応的に決定する手法を提案した．これにより性能向上を維持したまま，性能低下の発生を抑えることを両立する機構を実現した．

本研究の成果として，ここまでの提案手法を全て列挙したグラフを次の図 56 に示す．縦軸は LRU の IPC を 1 としたときの相対 IPC の増減をパーセントで示している．列数が多いため 2 段のグラフとしているが，上下段での条件の差はなく，共通して LRU からの相対 IPC で示している．このグラフから，それぞれの提案手法が理想手法である Stubborn-Manually-Best にどれだけ漸近しているかが見て取れる．

このように，シミュレータによる評価では本研究における 10 種の実行時ワークロード適応型の Stubborn キャッシュマネジメント手法のうち，LRU based Stubborn-HL-Reset が最良の結果を示し，LRU に対して最大 42.3%，幾何平均で最大 3.8%の性能向上を示した．Normal Stubborn における単純な Stubborn 戦略の適用と比較した場合

2.8%の性能向上となり，特に，7.4%性能低下していたワークロードでは性能低下を抑えて 0.1%の性能向上が得られるまで改善が見られるなど，提案の適応制御により性能を安定させる効果が確認できた．この結果より，本研究の成果として，長期保持による統計的なヒット数向上を可能とするような戦略である Stubborn 戦略とその適応的な拡張手法によって，大容量化した LLC を従来の手法とは異なるアプローチで活用することでメモリサブシステムによるデータ供給能力を高め，プロセッサの性能向上に寄与することを示した．


図 56 本研究における全提案手法の列挙

## 7.2. 今後の展望

### 7.2.1. マルチスレッド環境への適用

本研究の評価では，シングルコア，シングルスレッドのスーパスカラプロセッサにおけるキャッシュマネジメントを前提としたモデルが用いられている．マルチコア，マルチスレッド環境を想定した場合，本研究の提案手法は共有キャッシュとなることが多い LLC においても適用が可能である．その過程では，シングルスレッドでは考慮の対象とならなかったプロセス間，スレッド間のフェアネスを考慮した導入のための工夫が必要になる一方で，プロセス間で共通のデータを扱うことがあれば Stubborn 領域としての採択の優先度を上げるための情報として用いることができるといった展望がある．

### 7.2.2. 時間軸方向プリフェッチ

これまで述べたように，今後のキャッシュアルゴリズムが対処すべき対象は長期の再参照により生じるミスである．本研究では，長期の再参照間隔のアクセスによって生じるミスを削減する手法について，置き換えアルゴリズムの一戦略として実現する方法を模索してきた．これを実現する Stubborn 戦略は，従来の置き換えアルゴリズムが意識して保護してきた時間的局所性の高いキャッシュラインとは真逆の性質を持つキャッシュラインを長期間に渡って保持し続けることによって成立している．この手法では，キャッシュメモリの性質上スラッシングへの耐性を発揮するが，その性能はスラッシングの周期に依存してその何割かの固定されたアドレス帯へのアクセスでしかヒットしか得られない．例えば，一定の期間で使用するワーキングセットが 4MB で，Stubborn 領域によって保持できるデータ容量が 1MB であったとき，最大でも全体の 25%分しかキャッシュすることができない．

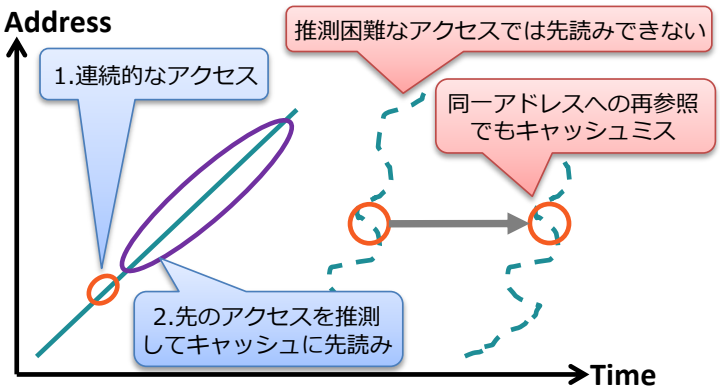



図 57 従来手法でのプリフェッチで対処できないパターン

一方で，長期の再参照間隔でアクセスのあるキャッシュラインは，時間的局所性は低くとも，一定時間の経過により同一のキャッシュラインへのアクセスがあるという意味では，従来の置き換えアルゴリズムで保護されるような時間的局所性の高いキャッシュラインと同様の空間的局所性を有する．Stubborn 領域に格納したデータは，再参照されるまでの長期間使用されない．一方で，再参照は各アドレスごとに一定間隔の周期をもって行われる．そこで，プリフェッチのトリガーとして，従来のプリフェッチのようなアドレス軸方向での空間的局所性を基にした参照予測（図 57）ではなく，時間軸方向での再参照間隔の履歴とその規則性を利用した再参照予測を行うことで，さらなるミス削減と性能向上の余地があるのではないかと期待する．この予想に基づくプリフェッチ手法を時間軸方向プリフェッチと名づけた．

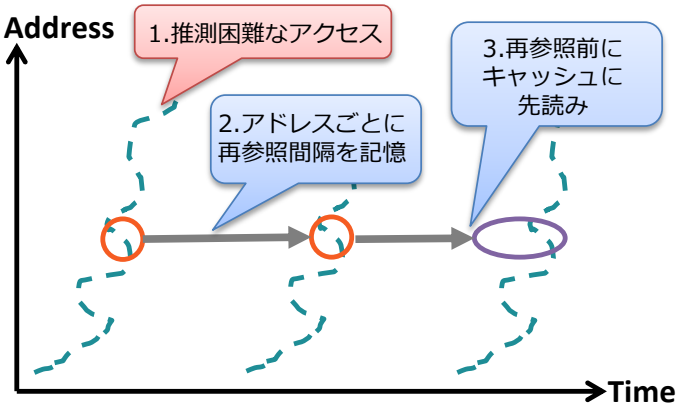



図 58 時間軸方向でのプリフェッチ

アクセスパターン傾向の調査の過程で，長期再参照は一定の時間間隔で行われていることがわかっている．例えば図 26，図 27 で示した実際のアクセスパターンでは，各アドレスについて再参照をそれぞれ約 1M 命令間隔，約 25M 命令間隔で繰り返している．つまり，アドレスと再参照間隔の対応関係から，そのアドレスについて次回の再参照のタイミングを図ることができる．これが時間軸方向プリフェッチの基本アイデアである．

このアイデアに基づいて，今後考えられる時間軸方向プリフェッチャの動作を図 58 に示す．長期再参照を起こすアドレスについては，その再参照間隔を記憶する，最後のアクセスから記憶した時間間隔の経過が近づいたタイミングでそのアドレスについてキャッシュに先読みすることでキャッシュミスを防ぐ．こうすることで，アドレス軸方向に推測困難なアクセスパターンであっても，時間軸方向に規則性を見いだしてプリフェッチに利用することで，再参照時にキャッシュヒットさせることができる．

長期再参照ミスへの対策において，長期間追い出さないことで実現している Stubborn 戦略とは対照的に，時間軸方向プリフェッチによりキャッシュラインを時分割して利用することで，Stubborn 領域に投じていたキャッシュ容量を超えるサイズのワーキングセットを扱うことができるようになることが期待できる．

本研究において長期再参照ミスの解消・削減が現在のキャッシュシステムにとっての課題であることを明らかにしたことで，このように長期の再参照間隔という時間軸方向の情報をプリフェッチのトリガーに利用する技術の研究，開発への挑戦に展望が開けた．このような新しいプリフェッチフレームワークは今後の大容量キャッシュの効率的な活用に不可欠な技術となると見据え，引き続き研究を行う．

# 謝辞

本博士論文は，筆者が東京大学大学院情報理工学系研究科電子情報学専攻在学中に坂井・入江研究室において，入江英嗣准教授の指導の下行った研究をまとめたものです.

本研究を進めるにあたって，入江英嗣准教授には本研究について，研究方針や手法の考察など，多くの助言や議論の機会をいただき，研究発表の推敲に至るまで，電気通信大学大学院所属時から現在まで通して絶えず丁寧な御指導を賜りました.

坂井修一教授には，学位審査における主査を引き受けていだたきました他，不自由なく研究できる環境の構築から，研究遂行における相談などを通じ様々な御指導を頂きました.

江崎浩教授，橋田浩一教授には学位審査における副査を引き受けていだたきました他，アドバイザ教員制度を通して御指導を頂きました．田浦健次郎教授には副査を引き受けていだたき，予備審査を通して御指導頂きました.

甲地弘幸君，渋江陽人君には研究において実装や評価，ツールの開発で助力いただき，国際会議発表，研究会発表においても共著者として協力いただきました.

中島拓真博士には，電気通信大学大学院在籍時から現在まで通して，議論を通じて本研究の応用先の展望や手法の拡張について多くの知見をいただき，また相互に知見を共有して広い分野で活用できるような研究を進めることができました.

事務補佐員の赤羽彩子さん，八木原晴水さんには，対外発表出張における手続きや研究設備の管理を行う上での事務などでお世話になりました.

また，坂井・入江研究室の学生の皆様，OB の宮永瑞紀氏にも，論文執筆や研究に関する議論，計算機資源の管理において大変お世話になりました.

皆様には，ここに深甚なる謝意を表します.

最後に，日常的な通学から長距離長期間の旅行まで，大学院生生活を支えてくれた GF-GC8 型インプレッサ WRX type-RA STi ver.VI Limited と，その製造元である富士重工業株式会社（現・株式会社 SUBARU）に感謝の意を表します.

# 参考文献


[1] T. Singh *et al.*, “3.2 Zen: A next-generation high-performance ×86 core,” in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 52–53.

[2] P. Conway, N. Kalyanasundharam, G. Donley, K. Lepak, and B. Hughes, “Cache Hierarchy and Memory Subsystem of the AMD Opteron Processor,” *IEEE Micro*, vol. 30, no. 2, pp. 16–29, Mar. 2010.

[3] “White Paper: Intel® Next Generation Microarchitecture (Nehalem),” p. 9.

[4] “Intel® Core™ i7-2640M Processor (4M Cache, up to 3.50 GHz) Product Specifications,” *Intel® ARK (Product Specs)*. [Online]. Available: https://ark.intel.com/products/53464/Intel-Core-i7-2640M-Processor-4M-Cache-up-to-3-50-GHz-.

[5] “Differentiating AM5K2E02 and AM5K2E04 SoCs from Alternate ARM® Cortex®-A15 Devic,” p. 6, 2014.

[6] C. Märtin, “Multicore Processors: Challenges, Opportunities, Emerging Trends,” p. 9, 2014.

[7] S. M. Khan, Y. Tian, and D. A. Jimenez, “Sampling Dead Block Prediction for Last-Level Caches,” in *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2010, pp. 175–186.

[8] P. Faldu and B. Grot, “Leeway: Addressing Variability in Dead-Block Prediction for Last-Level Caches,” in *2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2017, pp. 180–193.

[9] Newton, S. K. Mahto, S. Pai, and V. Singh, “DAAIP: Deadblock Aware Adaptive Insertion Policy for High Performance Caching,” in *2017 IEEE International Conference on Computer Design (ICCD)*, Boston, MA, 2017, pp. 345–352.

[10] D. Sanchez and C. Kozyrakis, “Vantage: Scalable and Efficient Fine-grain Cache Partitioning,” in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, New York, NY, USA, 2011, pp. 57–68.

[11] Bruce Hutton, “The Alpha Computer Architecture.” Jan-2007.

[12]Compaq Computer Corporation, “Alpha Architecture Handbook Version 4.” Oct-1998.

[13]M. K. Gowan, L. L. Biro, and D. B. Jackson, “Power considerations in the design of the Alpha 21264 microprocessor,” in *Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175)*, 1998, pp. 726–731.

[14]塩谷亮太, “プロセッサ シミュレータ「鬼斬弐」の設計と実装,” *先進的計算基盤システムシンポジウム 2009 ポスター*, 2009.

[15]K. Watanabe, H. Ichibayashi, M. Goshima, and S. Sakai, “Design of Processor Simulator ‘Onikiri,’” presented at the Advanced Computing Systems and Infrastructures (SACSIS), 2007, pp. 194–195.

[16]A. Jaleel, K. B. Theobald, S. C. Steely,Jr., and J. Emer, “High Performance Cache Replacement Using Re-reference Interval Prediction (RRIP),” in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, New York, NY, USA, 2010, pp. 60–71.

[17]S. Srinath, O. Mutlu, H. Kim, and Y. N. Patt, “Feedback Directed Prefetching: Improving the Performance and Bandwidth-Efficiency of Hardware Prefetchers,” in *IEEE 13th International Symposium on High Performance Computer Architecture, 2007. HPCA 2007*, 2007, pp. 63–74.

[18]“Standard Performance Evaluation Corporation: SPEC CPU 2006 Benchmark Suite.” [Online]. Available: https://www.spec.org/cpu2006/.

[19]富士通, “ベンチマークの概要 SPECcpu2006,” Sep. 2015.

[20]L. A. Belady, “A study of replacement algorithms for a virtual-storage computer,” *IBM Syst. J.*, vol. 5, no. 2, pp. 78–101, 1966.

[21]E. G. Coffman Jr. and P. J. Denning, *Operating Systems Theory*. Prentice Hall Professional Technical Reference, 1973.

[22]M. K. Qureshi, A. Jaleel, Y. N. Patt, S. C. Steely, and J. Emer, “Adaptive Insertion Policies for High Performance Caching,” in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, New York, NY, USA, 2007, pp. 381–391.

[23]A. Jaleel, W. Hasenplaugh, M. Qureshi, J. Sebot, S. Steely Jr., and J. Emer, “Adaptive Insertion Policies for Managing Shared Caches,” in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, New York, NY, USA, 2008, pp. 208–219.

[24] C.-J. Wu, A. Jaleel, M. Martonosi, S. C. Steely Jr., and J. Emer, "PACMan: Prefetch-aware Cache Management for High Performance Caching," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, New York, NY, USA, 2011, pp. 442–453.

[25] C.-J. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely Jr., and J. Emer, "SHiP: Signature-based Hit Predictor for High Performance Caching," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, New York, NY, USA, 2011, pp. 430–441.

[26] V. Young, C.-C. Chou, A. Jaleel, and M. K. Qureshi, "SHiP + + : Enhancing Signature-Based Hit Predictor for Improved Cache Performance," 2017.

[27] A. Jain and C. Lin, "Back to the Future: Leveraging Belady's Algorithm for Improved Cache Replacement," 2016.

[28] A. Jain and C. Lin, "Hawkeye : Leveraging Belady ' s Algorithm for Improved Cache Replacement," 2017.

[29] A. J. Smith, "Sequential Program Prefetching in Memory Hierarchies," *Computer*, vol. 11, no. 12, pp. 7–21, Dec. 1978.

[30] J. W. C. Fu, J. H. Patel, and B. L. Janssens, "Stride Directed Prefetching in Scalar Processors," in *Proceedings of the 25th Annual International Symposium on Microarchitecture*, Los Alamitos, CA, USA, 1992, pp. 102–110.

[31] K. J. Nesbit and J. E. Smith, "Data Cache Prefetching Using a Global History Buffer," in *Proceedings of the 10th International Symposium on High Performance Computer Architecture*, Washington, DC, USA, 2004, pp. 96–96.

[32] Y. Ishii, M. Inaba, and K. Hiraki, "Access Map Pattern Matching Prefetch: Optimization Friendly Method," presented at the Workshop on JILP Data Prefetching Championship, 2009, p. 5.

[33] Y. Ishii, M. Inaba, and K. Hiraki, "Access Map Pattern Matching for Data Cache Prefetch," in *Proceedings of the 23rd International Conference on Supercomputing*, New York, NY, USA, 2009, pp. 499–500.

[34] X. Zhuang and H.-S. Lee, "A hardware-based cache pollution filtering mechanism for aggressive prefetches," in *2003 International Conference on Parallel Processing, 2003. Proceedings.*, 2003, pp. 286–293.

[35]入江英嗣, 本城剛毅, 平木敬, “動的推定によるプリフェッチ量最適化,” 情報処理学会論文誌 コンピューティングシステム, vol. 3, no. 3, pp. 56–66, Sep. 2010.
[36]H. Irie, T. Miyoshi, G. Honjo, K. Hiraki, and T. Yoshinaga, “Using Cacheline Reuse Characteristics for Prefetcher Throttling,” *IEICE Trans. Inf. Syst.*, vol. E95-D, no. 12, pp. 2928–2938, Dec. 2012.
[37]渋江陽人, 野村隼人, 入江英嗣, 坂井修一, “ストライドアクセスの階層構造に着目したフェーズ検出,” *電子情報通信学会技術研究報告 IEICE Tech. Rep. 信学技報*, vol. 116, no. 510, pp. 9–14, Mar. 2017.

# 著者発表論文

計算機アーキテクチャ研究に関するもの


[1] K. Hagiwara, T. Hayashi, S. Kawasaki, F. Arakawa, O. Endo, H. Nomura, A. Tsukamoto, D. Nguyen, B. Nguyen, A. Tran, H. Hyunh, I. Kudoh, and C.-K. Pham, “A Two-stage-pipeline CPU of SH-2 Architecture Implemented on FPGA and SoC for IoT, Edge AI and Robotic Applications,” The 21th International Symposium on Low-Power and High-Speed Chips (COOL Chips 21), Yokohama, vol.XII–3, pp.1–19, Apr. 2018.

[2] 渋江陽人, 野村隼人, 入江英嗣, 坂井修一, “ストライドアクセスの階層構造に着目したフェーズ検出,” 電子情報通信学会技術研究報告 = IEICE technical report : 信学技報, vol.116, no. 510, pp.9–14, Mar. 2017.

[3] 野村隼人, 甲地弘幸, 入江英嗣, 坂井修一, “置き換えをしない置き換えアルゴリズム,” The 13th IEEE Transdisciplinary-Oriented Workshop for Emerging Researchers, 東京, p.31, Dec. 2016.

[4] H. Nomura, H. Katchi, H. Irie, and S. Sakai, “”Stubborn” strategy to mitigate remaining cache misses,” 2016 IEEE 34th International Conference on Computer Design (ICCD), pp.388–391, Oct. 2016.

[5] 村田篤志, 野村隼人, 吉見真聡, 吉永努, 入江英嗣, “3 次元積層プロセッサ向けフロアプランナの可視化,” 電子情報通信学会技術研究報告 = IEICE technical report : 信学技報, vol.115, no. 243, pp.63–65, Oct. 2015.

[6] H. Nomura, T. Nakajima, M. Yoshimi, T. Yoshinaga, and H. Irie, “Stubborn Cache: A Novel Strategy for Repeating Thrashing Access Patterns,” The 18th International Symposium on Low-Power and High-Speed Chips (COOL Chips XVIII), Yokohama, p.19, Apr. 2015.

[7] 野村隼人, 力翠湖, 吉見真聡, 吉永努, 入江英嗣, “動的推定によるキャッシュパーティショニング最適化,” 研究報告計算機アーキテクチャ（ARC）, vol.2014, no. 2, pp.1-6, Jul. 2014.

[8] 野村隼人, 嶋田創, 小林良太郎, “割り込み要求線バンドルによる高信頼割り込みインタフェースの評価,” 電子情報通信学会技術研究報告 = IEICE technical report : 信学技報, vol.113, no. 282, pp.7–12, Dec. 2013.

[9] H. Nomura, H. Shimada, and R. Kobayashi, “Preliminary Discussion of Dependable Interrupt Management for Microcontrollers,” The 16th International Symposium on Low-Power and High-Speed Chips (COOLChips XVI), Yokohama, p.19, Apr. 2013.

その他の研究に関するもの

[10] 野村隼人, 中川瑛, “テクノロジーツリーの構築,” The 14th IEEE Transdisciplinary-Oriented Workshop for Emerging Researchers, 東京, p.52, Nov. 2017.
[11] 野村隼人, “物品捜索支援システムおよび物品管理システム,” 特許 608628801, Mar. 2017.
[12] 土門憲司, 野村隼人, 吉見真聡, 吉永努, 入江英嗣, “RGB-D センサと学習による運転姿勢検知,” 電子情報通信学会技術研究報告 = IEICE technical report : 信学技報, vol.115, no. 243, pp.67–69, Oct. 2015.
[13] 高羽雄太, 野村隼人, 山際康貴, “子どもたちはロボットをどのようなものとして捉えるのか ーロボットの内なる視点から探るー,” 第 2 回 リサーチフェスタ 2012, 茨城, p.38, Aug. 2012.
[14] 野村隼人, 村田充利, 森徹, 謝孟春, “RFID を用いた物品探知機と共用可能な物品管理システムの開発,” 第 19 回 電気学会関西支部 高専卒業研究発表会, 大阪, pp.25-26, Mar. 2012.
[15] 野村隼人, 村田充利, 森徹, 謝孟春, “RFID を用いた家庭用物品管理システムの開発,” 第 17 回高専シンポジウム in 熊本, 熊本, p.418, Jan. 2012.