オペレーティングシステム

#12 主記憶管理:ページ置き換え方式

復習：主記憶アクセスの局所性

- 主記憶アクセスの特徴 ＝ 局所性

```
for(i=0; i<n; i++){
        sum += a[i];
}
```

- 空間的局所性
  - あるアドレス・ページにアクセスが発生した場合，次はその近くのアドレス・ページにアクセスされる可能性が高い
- 時間的局所性
  - あるアドレス・ページにアクセスが発生した場合，近いうちには同じアドレス・ページにアクセスされる可能性が高い

復習：仮想記憶処理の実装

■ 仮想記憶処理のオーバヘッド削減

- ページフォルトの発生回数を減らしたい
  - → プリページング
    - 必要となりそうなページを前もってスワップイン

最近アクセスされたページの
近くにあるページ

空間的局所性
から

- ページフォルトの発生回数を減らしたい
  - → スワップアウト対象ページの効率的選択
    - この先，最も参照されなさそうなページをスワップアウト

最近アクセスされていない
(Least Recently Usedな)ページ

時間的局所性
から

今日の内容

- 仮想記憶処理のオーバヘッド削減
  - ページフォルトの発生回数を減らしたい
    - → プリページング
      - 必要となりそうなページを前もってスワッ
    - 最近アクセスされたページの近くにあるページ

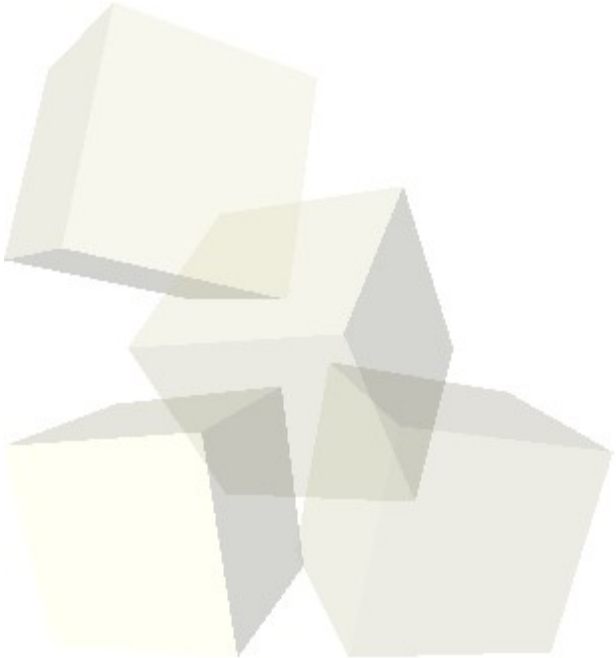
ページの置き換え方式についてより詳しく

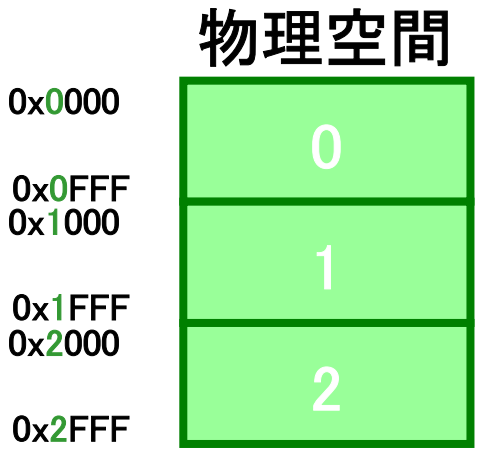
  - ページフォルトの発生回数を減らしたい
    - → スワップアウト対象ページの効率的選択
      - この先，最も参照されなさそうなページをスワップアウト
    - 最近アクセスされていない(Least Recently Usedな)ページ

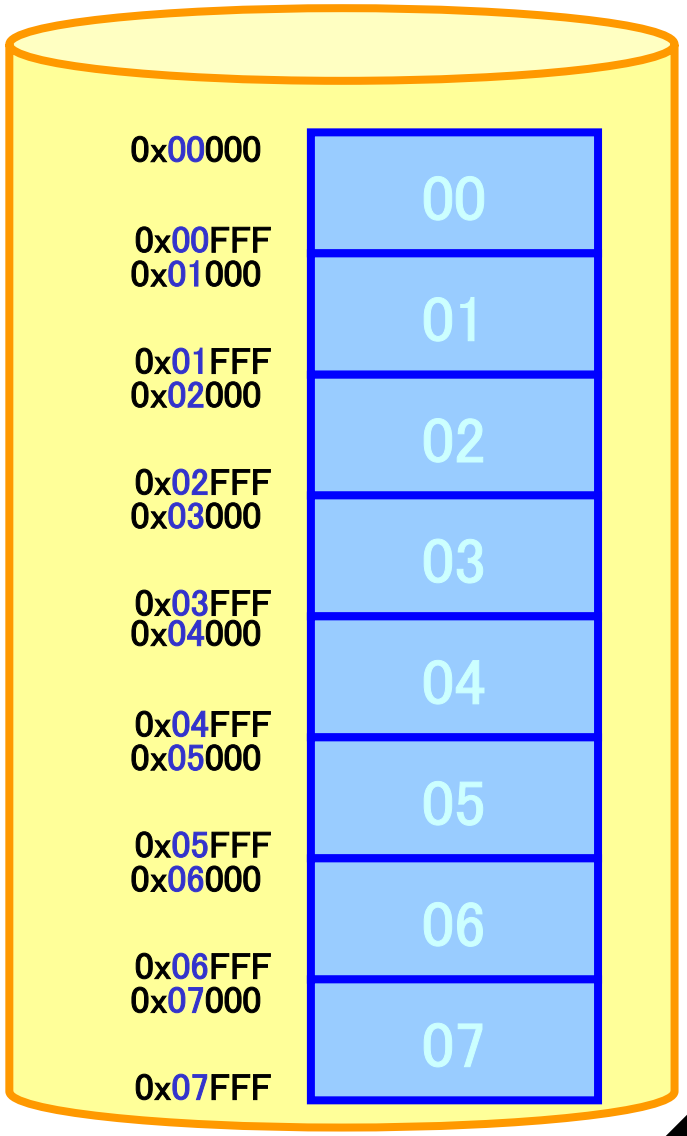
12.1
静的ページ置き換え方式

例

- これ以降，以下の例を使って説明
  - ページフレーム数
    （物理空間）：3 (0～2)
  - ページ数
    （仮想空間）：8 (00～07)







例

- あるプロセスが，
  以下の参照順でアクセスした場合

  - 00 → 01 → 02 → 03 →
    00 → 01 → 04 → 00 →
    01 → 02 → 03 → 04

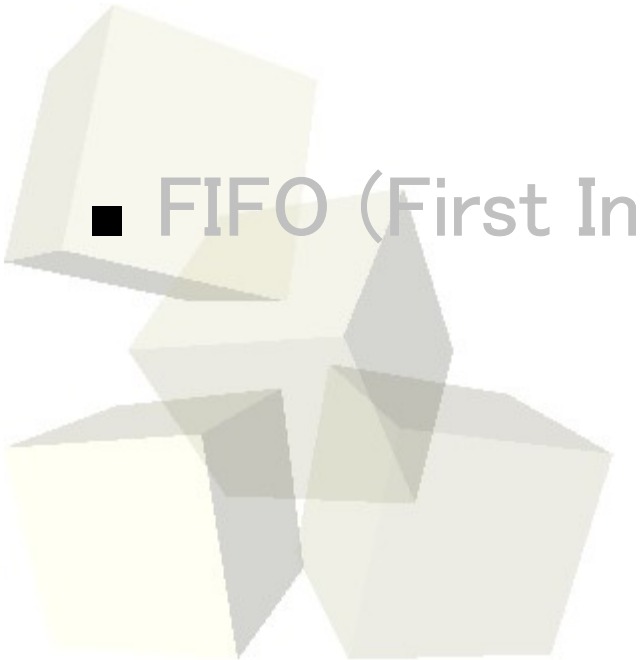
- さまざまなアルゴリズムにおける
  ページフォルト発生回数を見る

静的ページ置き換え方式

- 最適アルゴリズム
- LRU (Least Recently Used)
- LFU (Least Frequently Used)
- FIFO (First In First Out)

静的ページ置き換え方式

- 最適アルゴリズム
- LRU (Least Recently Used)
- LFU (Least Frequently Used)
- FIFO (First In First Out)



最適アルゴリズム

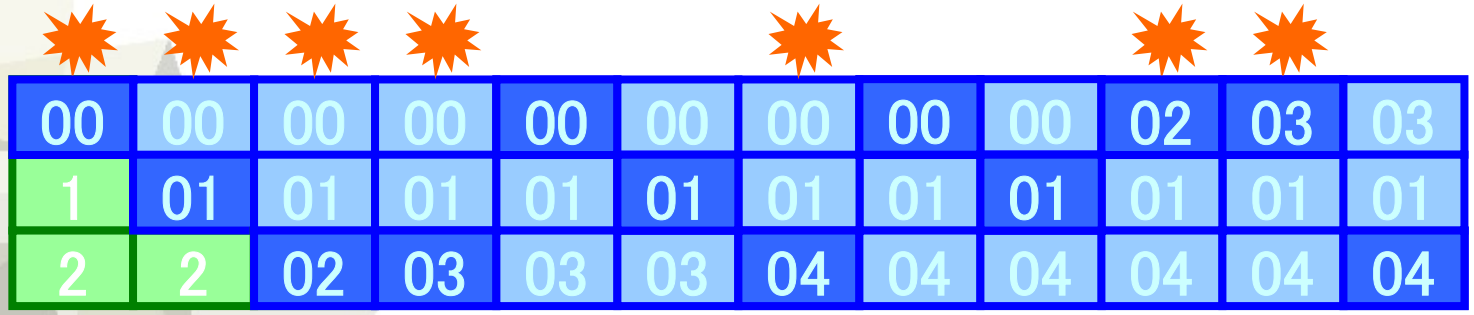
前もって参照列が分かっていると仮定

- 参照順
  - 00 → 01 → 02 → 03 →
    00 → 01 → 04 → 00 →
    01 → 02 → 03 → 04

ページフォルト
回数

0 7

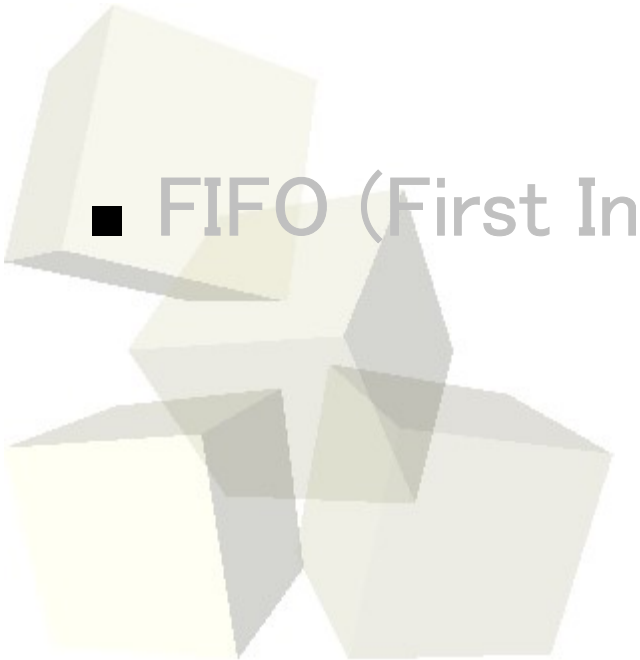
- ページの変遷



時刻

静的ページ置き換え方式

- 最適アルゴリズム
- LRU (Least Recently Used)
- LFU (Least Frequently Used)
- FIFO (First In First Out)



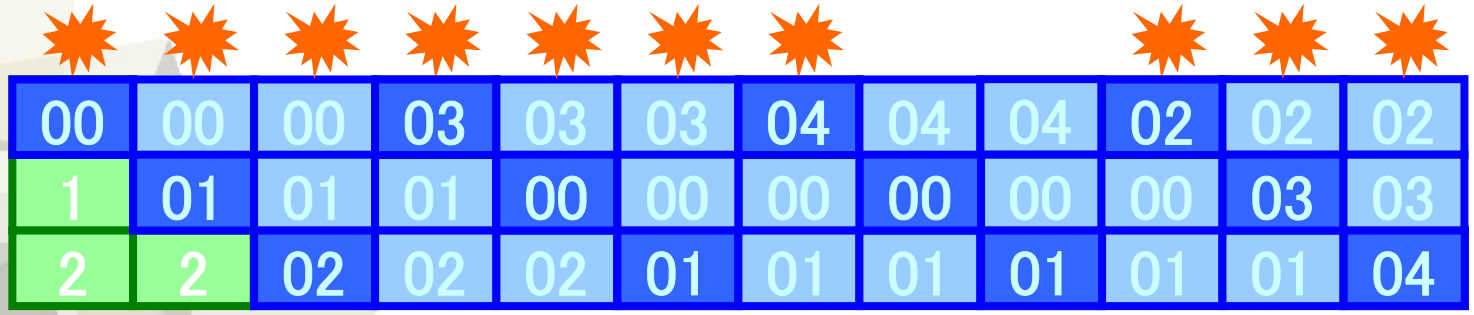
LRU

- 参照順
  - 00 → 01 → 02 → 03 →
    00 → 01 → 04 → 00 →
    01 → 02 → 03 → 04

ページフォルト
回数

1 9

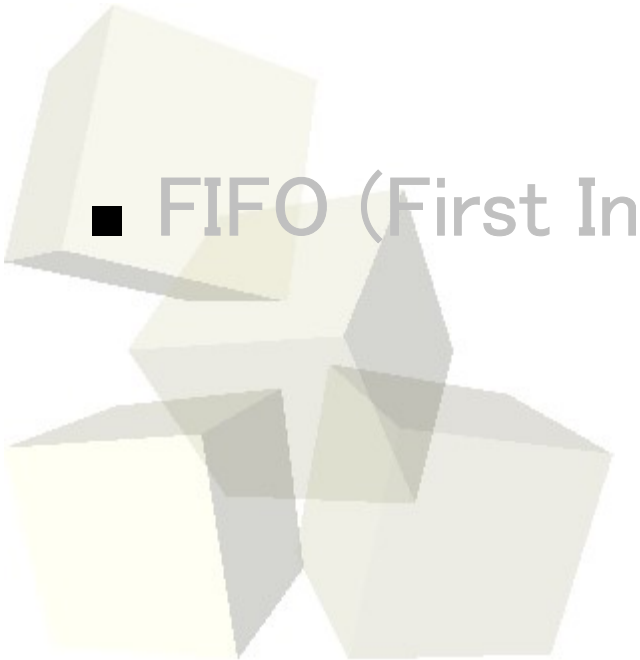
- ページの変遷



時刻

静的ページ置き換え方式

- 最適アルゴリズム
- LRU (Least Recently Used)
- LFU (Least Frequently Used)
- FIFO (First In First Out)



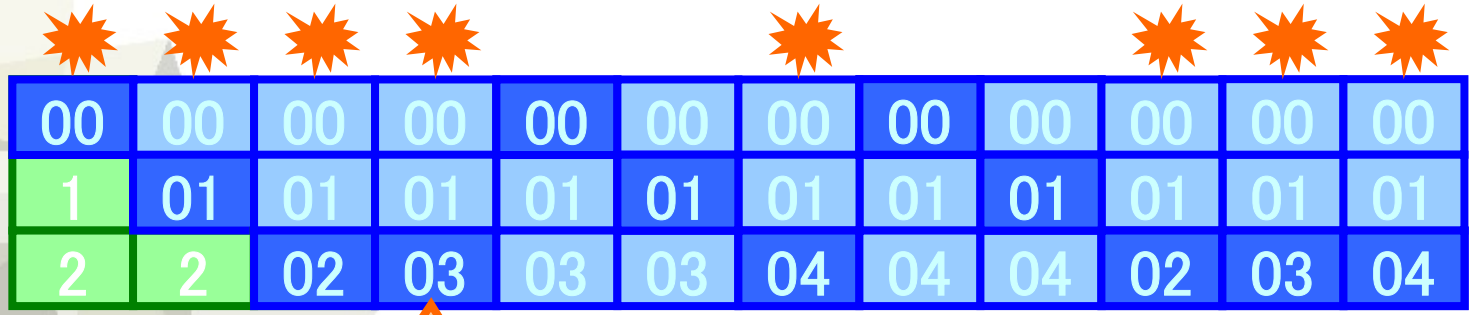
LFU

- 参照順
  - 00 → 01 → 02 → 03 →
    00 → 01 → 04 → 00 →
    01 → 02 → 03 → 04

ページフォルト
回数

0 8

- ページの変遷



静的ページ置き換え方式

- 最適アルゴリズム
- LRU (Least Recently Used)
- LFU (Least Frequently Used)
- FIFO (First In First Out)

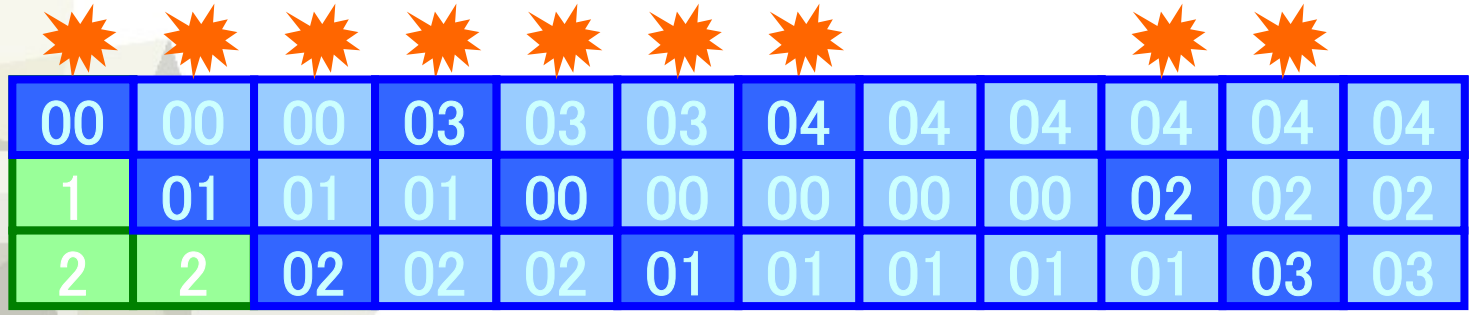
FIFO

- 参照順
  - 00 → 01 → 02 → 03 →
    00 → 01 → 04 → 00 →
    01 → 02 → 03 → 04

ページフォルト
回数

0 9

- ページの変遷

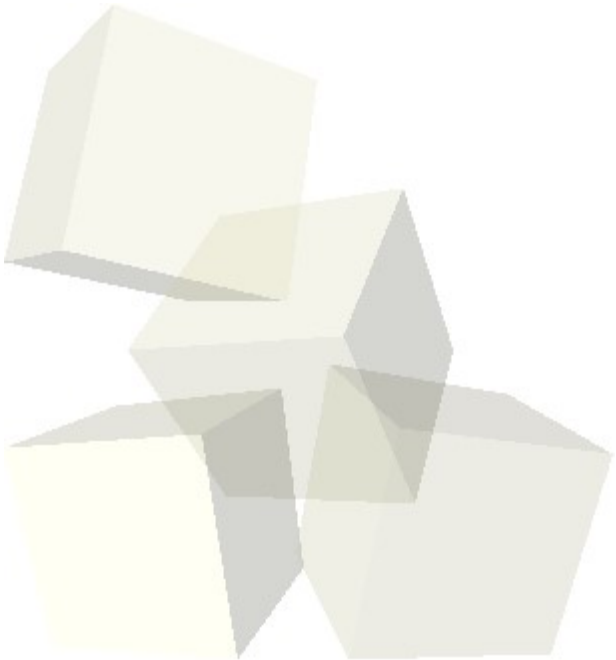


時刻

まとめ：静的ページ置き換え方式

- 最適アルゴリズム
  - 最小ページフォルト回数を確認するために紹介
  - 実現不可能
- LRU (Least Recently Used)
  - 直近のアクセス時刻が最も過去のものを
    スワップアウト
- LFU (Least Frequently Used)
  - スワップイン以降のアクセス頻度が最も低いものを
    スワップアウト
- FIFO (First In First Out)
  - スワップイン時刻が最も過去のものをスワップアウト

12.2
Beladyの例外



Beladyの例外

- 置き換えアルゴリズム
  - ページ参照列によっては得手・不得手はある
  - が，全てそれなりに「リクツ」の通ったアルゴリズム
  - なので，だいたいはうまくいきそう…

  - ページフレーム数が変化したとき
    思わぬ性能変化を引き起こすアルゴリズムが…

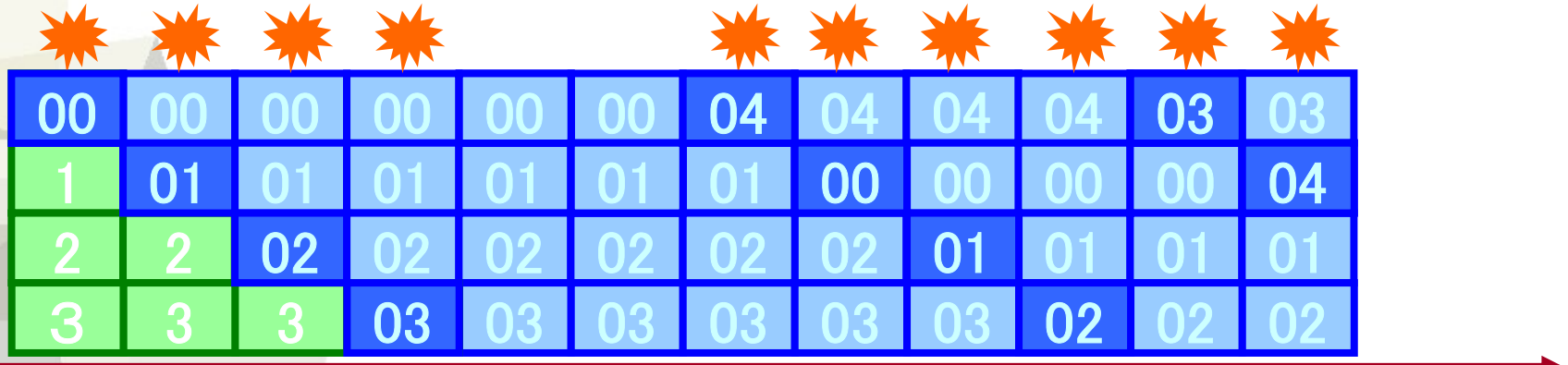
FIFO（ページフレーム数：4）

- 参照順
  - 00 → 01 → 02 → 03 →
    00 → 01 → 04 → 00 →
    01 → 02 → 03 → 04

ページフォルト
回数

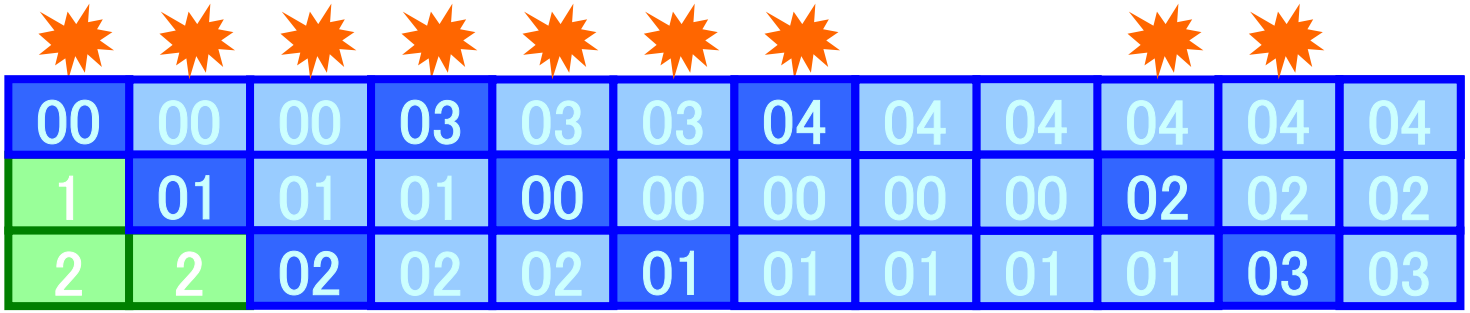
1 9

- ページの変遷



FIFOにおけるBeladyの例外

- ページフレーム数：3

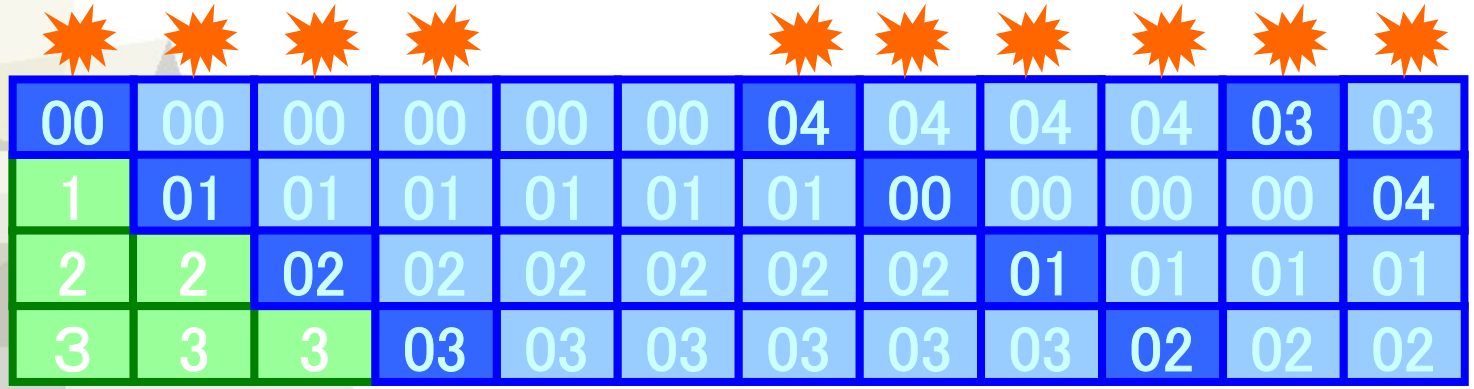


ページフォルト回数

0 9

ページフレーム数を増やすことで
ページフォルト回数増加!

- ページフレーム数：4



時刻

ページフォルト回数

1 0

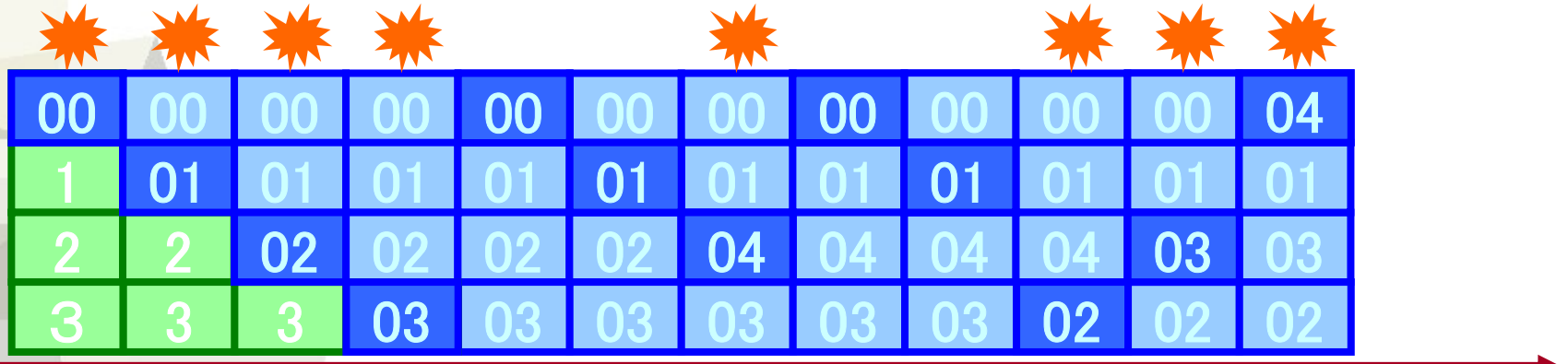
LRU（ページフレーム数：4）

- 参照順
  - 00 → 01 → 02 → 03 →
    00 → 01 → 04 → 00 →
    01 → 02 → 03 → 04

ページフォルト
回数

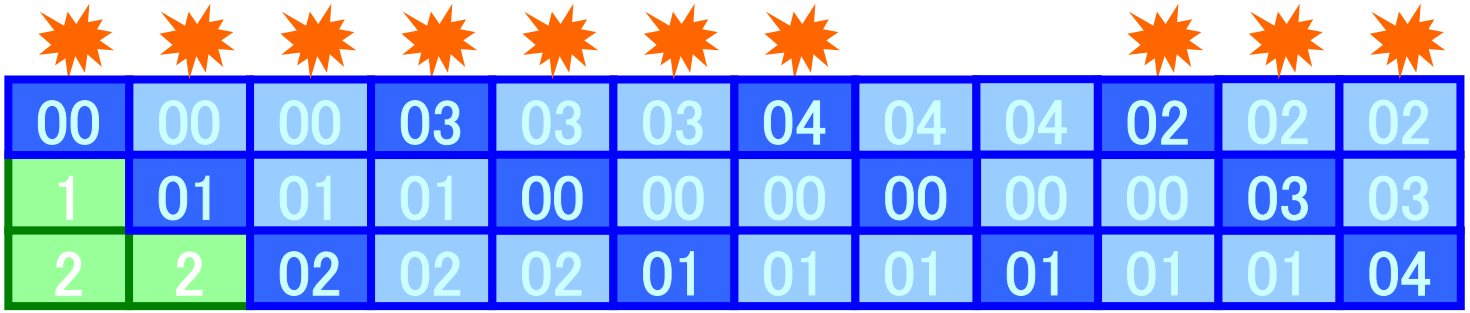
0 8

- ページの変遷



LRUの場合

- ページフレーム数：3



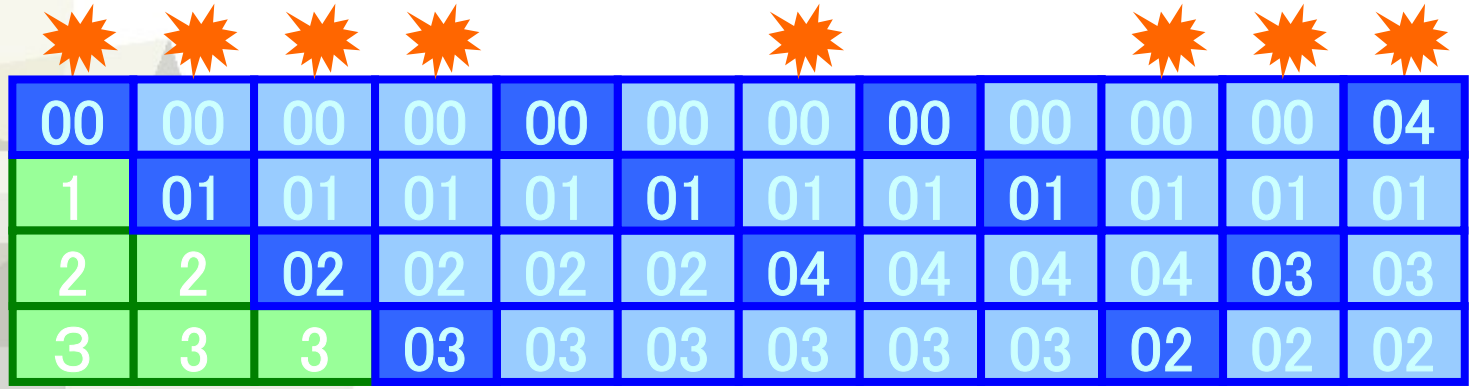
00	00	00	03	03	03	04	04	04	02	02	02
1	01	01	01	00	00	00	00	00	00	03	03
2	2	02	02	02	01	01	01	01	01	01	04

ページフォルト回数

10

ページフレーム数を増やすことで
ちゃんとページフォルト回数減少

- ページフレーム数：4



00	00	00	00	00	00	00	00	00	00	00	04
1	01	01	01	01	01	01	01	01	01	01	01
2	2	02	02	02	02	04	04	04	04	03	03
3	3	3	03	03	03	03	03	03	02	02	02

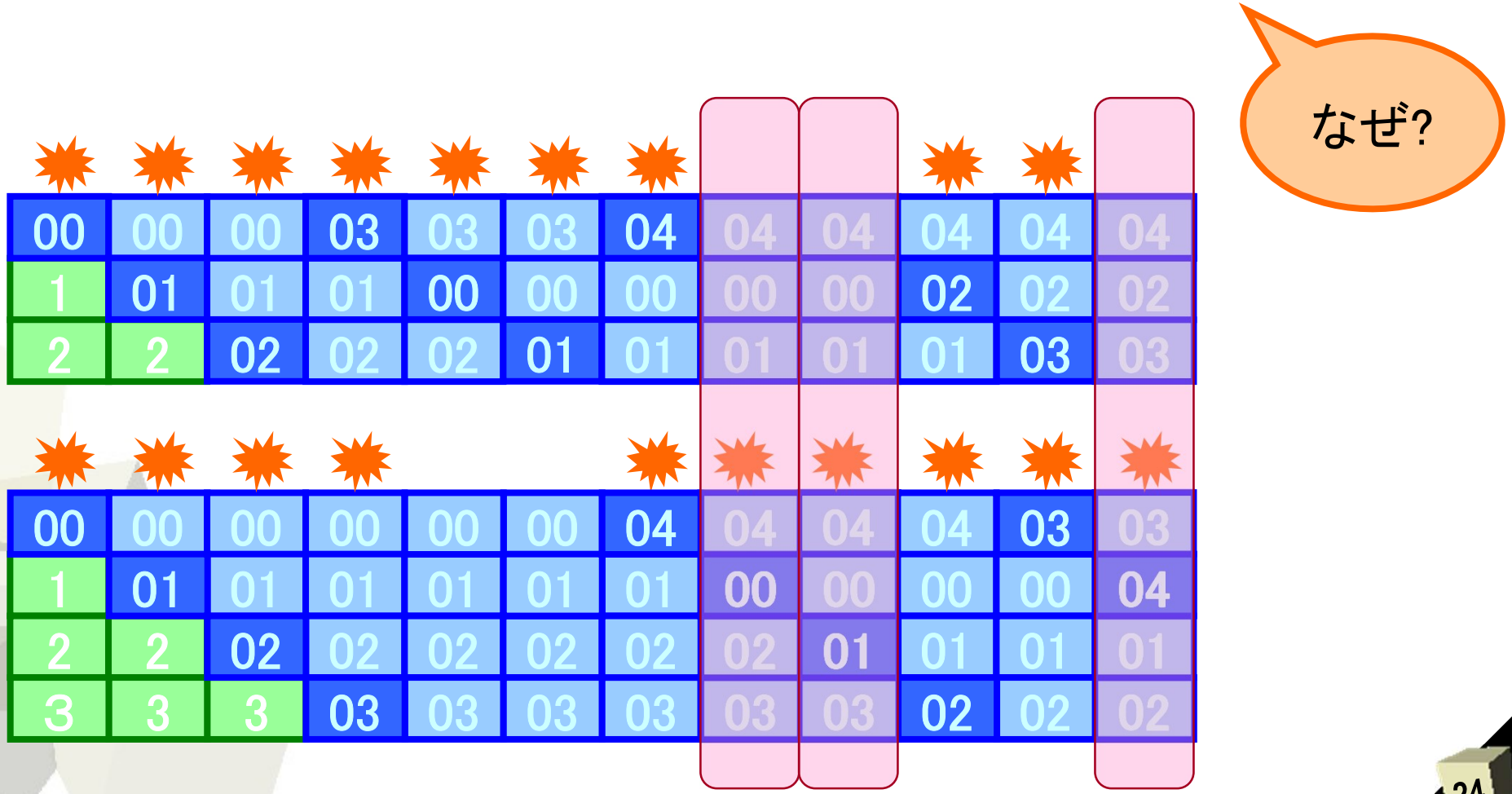
ページフォルト回数

08

時刻

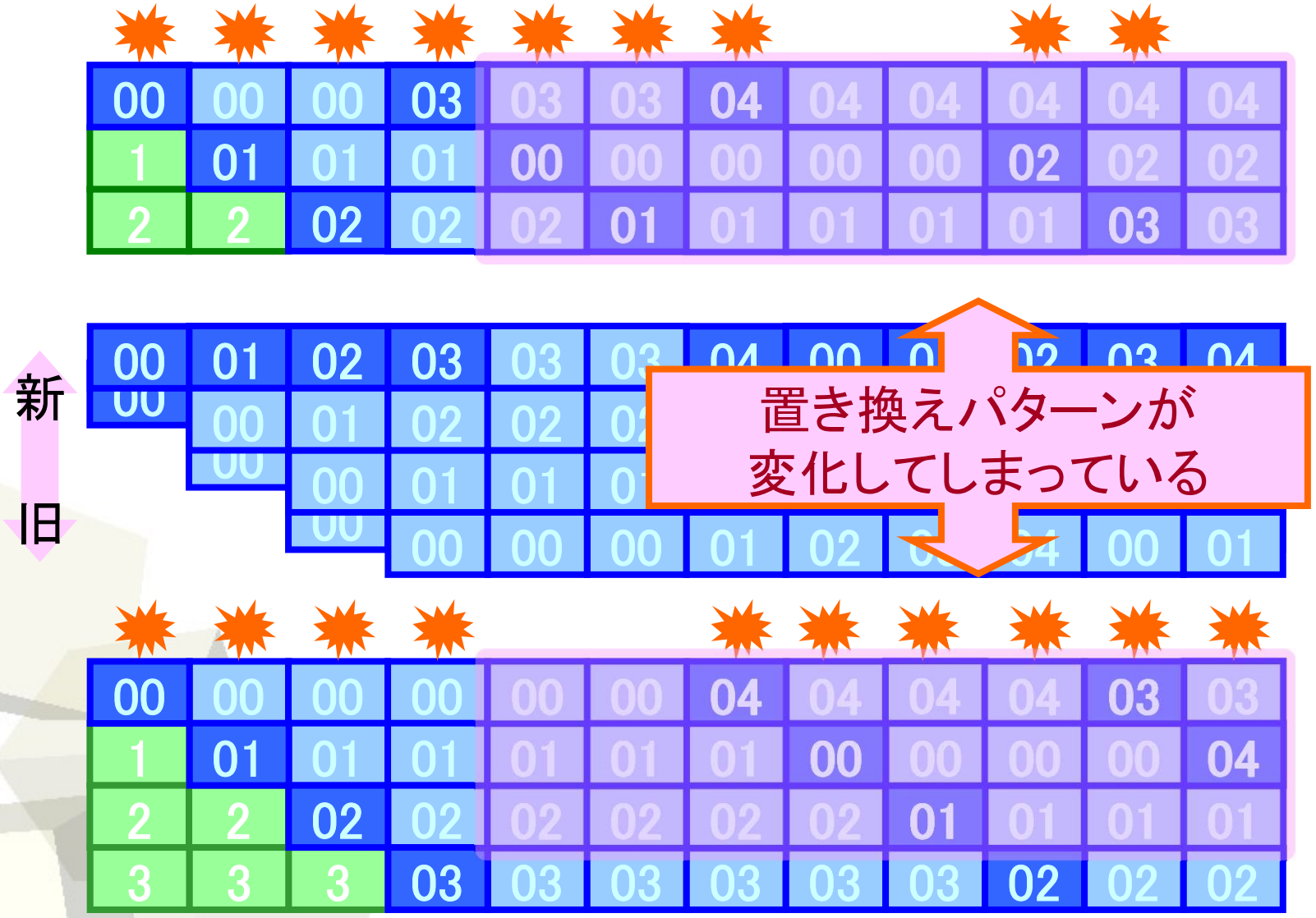
Beladyの例外の原因

- ページフォルトが増えるということは…
  - フレーム数3のときには発生しなかった
    ページフォルトがフレーム数4のときに発生している



ページの「あたらしさ」の変化

- ページを新しい順に並べてみる



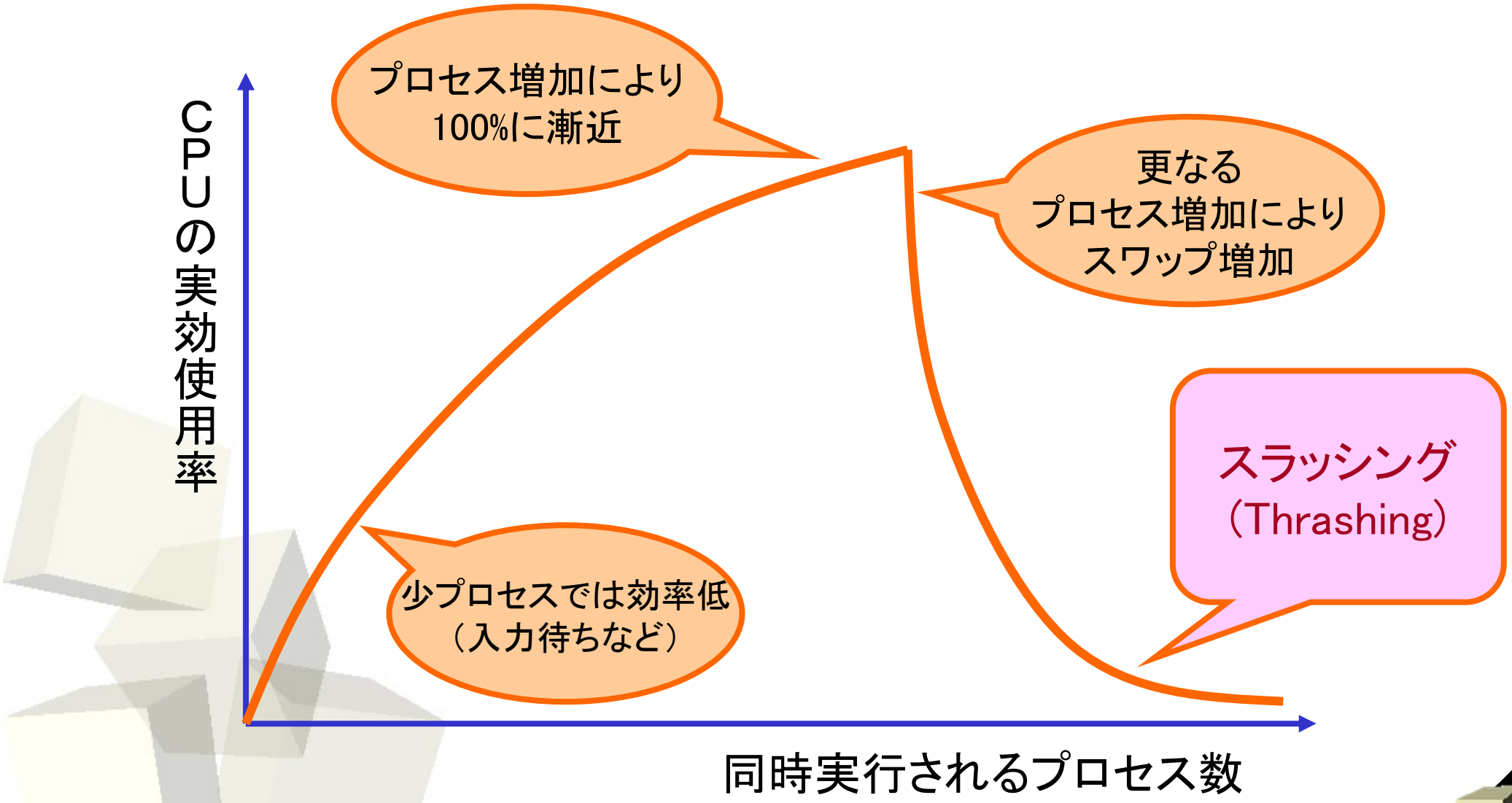
参考：LRUの場合

- ページを新しい順に並べてみる

12.2.1
スラッシング

12.3
動的ページ置き換え方式

実行プロセス数と実効処理能力



スラッシング（Thrashing）

定義

- 物理ページへの大量の要求により，
  CPUがページアウト/インのための処理に終始し，
  プロセスの動作を妨げている状態

原因

- 非常に多くのプロセスを並行動作させようとした
- 非常に大きな記憶領域を要求するプロセスを
  動作させようとした

スラッシングの軽減法

- ワーキングセット
  - 過去の単位時間にアクセスされたページの集合
  - if (ページフレーム数 ＜ ワーキングセットの大きさ)
    - → 頻繁なスワップイン・スワップアウトが発生
    - → スラッシング

ページフレーム数が
ワーキングセットの大きさと
同じだけあればよい

- ワーキングセット法
  - ワーキングセットの大きさと同じだけの
    ページフレームをプロセスに割り当てることで
    スラッシングを回避

ワーキングセット法の近似

- ワーキングセットの大きさ
  = プロセスに割り当てるページ数
  - ワーキングセットを調べるのはコスト膨大
  - なんらかの方法で近似

この情報を
静的置換えアルゴリズムに
加えて使う

- ページフォルト発生の平均間隔
  - 大きい場合（頻度小）
    → プロセスに与えられているページフレームは比較的十分
  - 小さい場合（頻度大）
    → プロセスには十分なページフレームが与えられていない

ページフォルト平均間隔＋LRU

- ページフォルト平均間隔 ＋ LRU
  - ワーキングセット法の近似とLRUの組み合わせ
  - プロセスに割当てるページフレーム数を動的に変更
    - → プロセスのワーキングセットは実行に伴い変化するため
- アルゴリズム
  - ページフォルトの平均間隔を計算
  - 平均間隔が（ある値より）小さい（高頻度）場合
    - → プロセスに与えるページフレーム数を増やす
      （次回ページフォルト時スワップアウトせずスワップイン）
  - 平均間隔が（ある値より）大きい（低頻度）場合
    - → プロセスに与えるページフレーム数を減らす
      （すぐにLRUに基づき強制的にスワップアウト）

まとめ：静的ページ置き換え方式

- LRU (Least Recently Used)
  - 直近のアクセス時刻が最も過去のものを
    スワップアウト
- LFU (Least Frequently Used)
  - スワップイン以降のアクセス頻度が最も低いものを
    スワップアウト
- FIFO (First In First Out)
  - スワップイン時刻が最も過去のものをスワップアウト

まとめ

ページ置き換え方式

- ページ参照列によっては得手・不得手がある
  - → 特定の参照列における性能だけを見て，
    一概にどれがよいとは言えない

Beladyの例外

- ページフレーム数をふやしたとき
  ページフォルトが増加してしまう現象
- 原因：
  ページフレーム数により
  置き換えパターンが変化してしまうアルゴリズム
  - → 例）FIFO

まとめ

- スラッシング
  - スワップ処理が多発し，CPUの本来の仕事である
    プロセス処理が妨げられる状態
    - → CPUの実行利用率が激減

- ワーキングセット法
  - スラッシングの緩和
  - 過去の単位時間においてアクセスされたページ群
    （ワーキングセット）のページ数と同じだけの
    ページフレームをプロセスに割り当てる

まとめ

- ページフォルト平均間隔
  - 近似的にワーキングセット法を実現する方法
    - → 実行時にワーキングセットを知ることは困難であるため