オペレーティングシステム

第11回

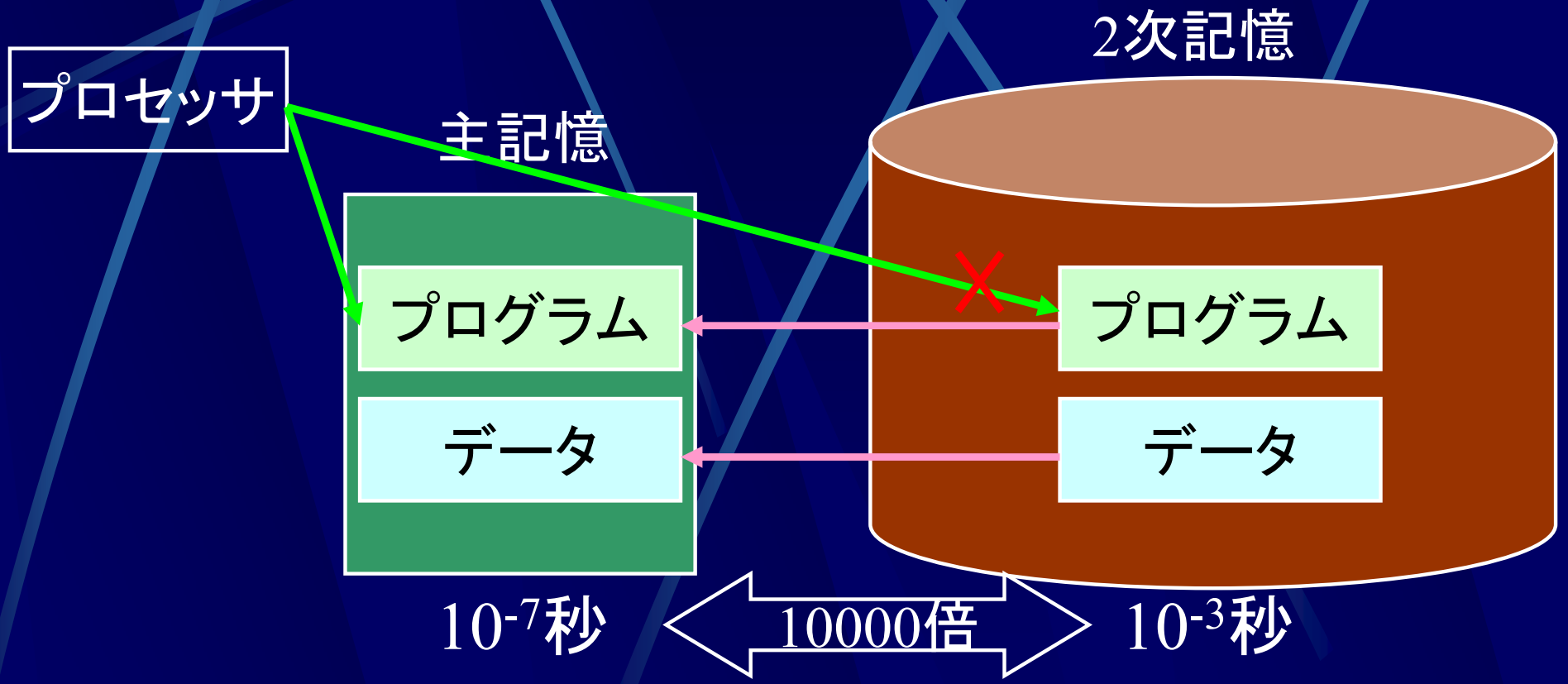
仮想記憶管理(3)

http://www.info.kindai.ac.jp/OS

E号館3階E-331 内線5459

takasi-i@info.kindai.ac.jp

主記憶と2次記憶



プロセッサは2次記憶を直接読むことはできない

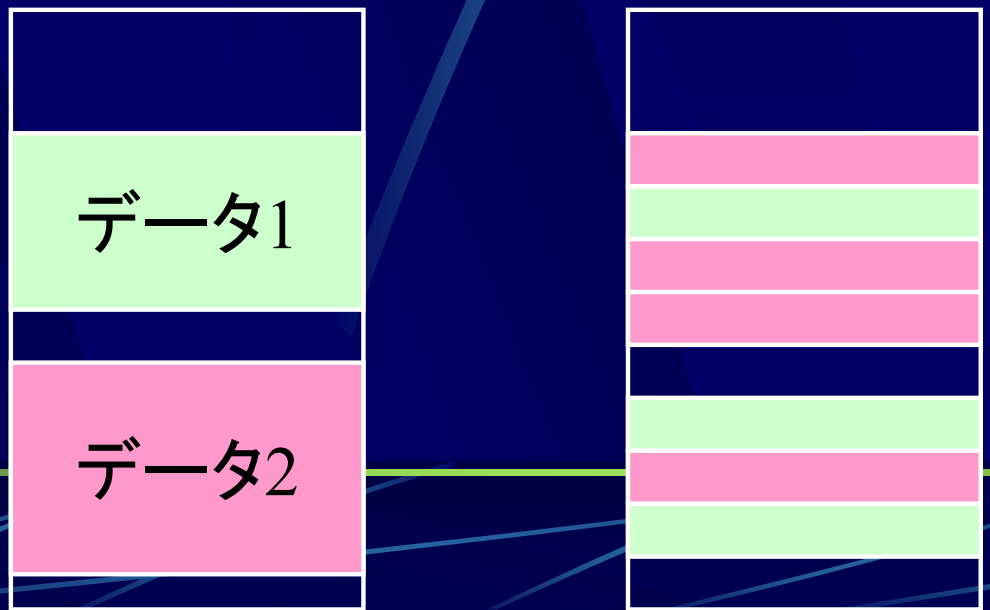
使用するプログラム, データは主記憶上にコピー

メモリ管理技法

- メモリ管理技法
  - 割り付け技法(placement)
    - プログラム, データのメモリ上への割り付け位置を決定
  - フェッチ技法(fetch)
    - プログラム, データを2次記憶から主記憶への読み込み時期を決定
  - 置き換え技法(replacement)
    - 空き領域作成のために2次記憶に追い出すデータの決定

割り付け技法(placement)

- 割り付け技法
  - 連続割り付け(contiguous allocation)
    - プログラム, データをメモリ上の連続した領域に置く
  - 非連続割り付け(noncontiguous allocation)
    - プログラム, データをメモリ上に分割して置く



割り付け技法

<table>
<tr><td rowspan="3">連続割り付け
(contiguous allocation)</td><td>単一連続割り付け
(single partition allocation)</td><td>単一ユーザ</td></tr>
<tr><td>固定区画割り付け
(static partition allocation)</td><td rowspan="4">複数ユーザ</td></tr>
<tr><td>可変区画割り付け
(dynamic partition allocation)</td></tr>
<tr><td rowspan="2">非連続割り付け
(noncontiguous allocation)</td><td>ページング
(paging)</td></tr>
<tr><td>セグメンテーション
(segmentation)</td></tr>
</table>

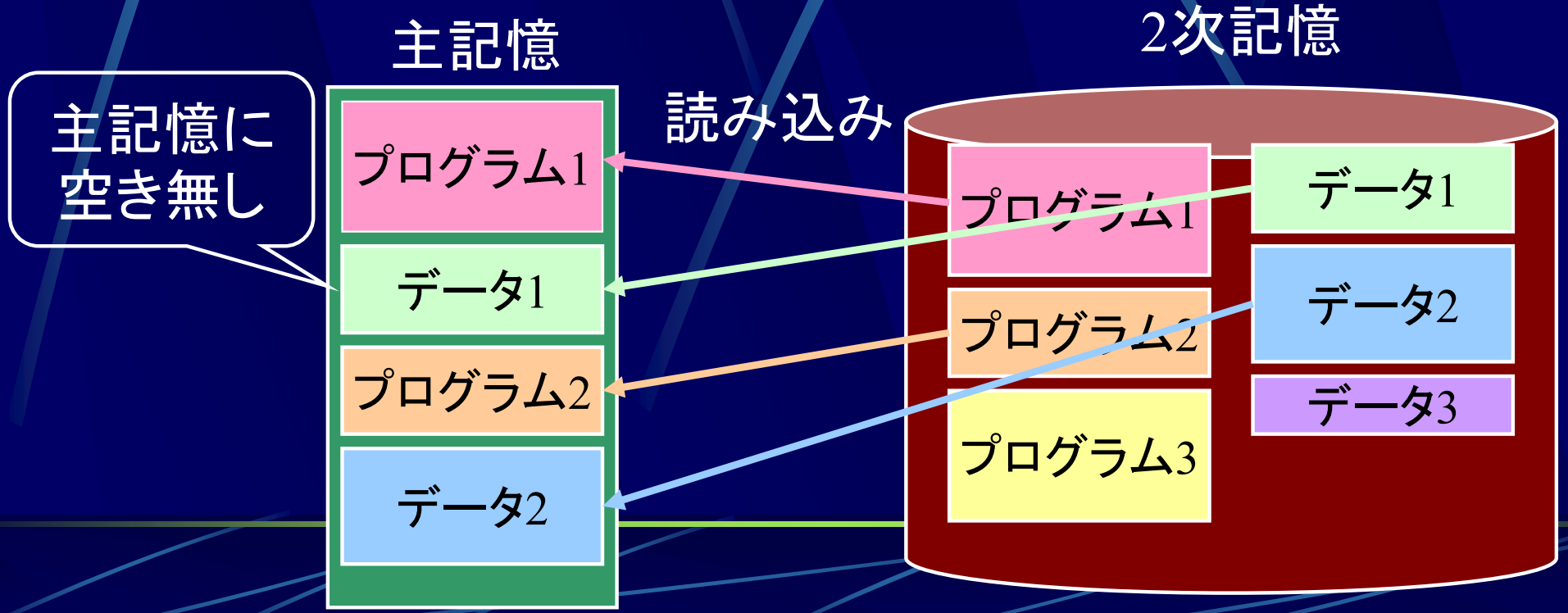
フェッチ技法(fetch)

- フェッチ技法
  - 要求時フェッチ(demand fetch)
    - プログラムが参照したときにデータを読み込む
  - プリフェッチ(prefetch)
    - 参照前に予めデータを読んでおく

フェッチ技法	ページング	食材
要求時フェッチ	必要としたときにページイン	毎回食事前に購入
プリフェッチ	必要なページを予測して予めページイン	1週間分のメニューを予測して週末に購入

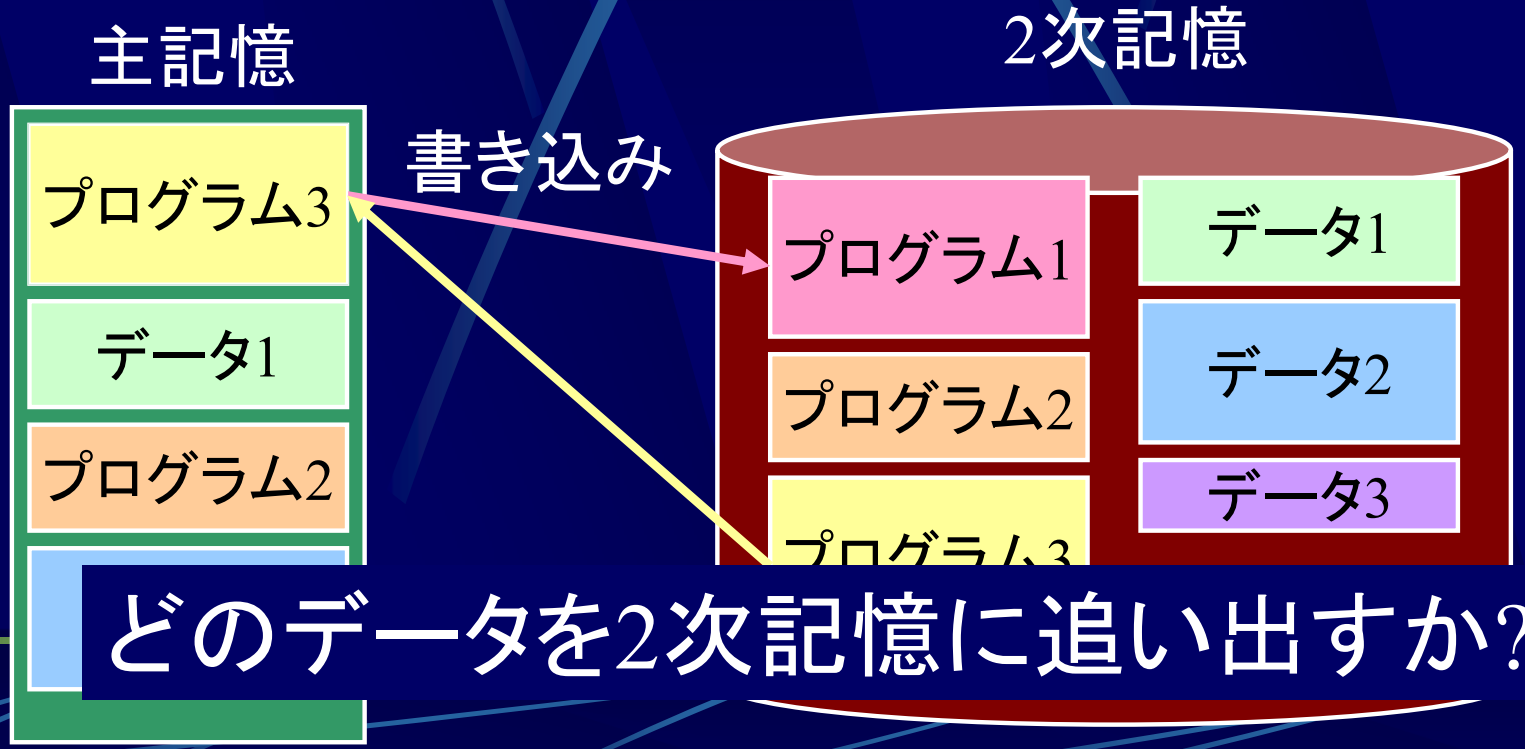
置き換え技法(replacement)

- 置き換え技法(replacement)
  - 空き領域作成のために2次記憶に追い出すデータの決定



置き換え技法(replacement)

- 置き換え技法(replacement)
  - 空き領域作成のために2次記憶に追い出すデータの決定



置き換え技法

- 2次記憶へのアクセスは時間がかかる
  (主記憶へのアクセスの約10000倍)

スワップ操作 (スワップイン，スワップアウト)

- 可能な限り低頻度に
- 可能な限り低コストに

ページングの動作

主記憶上に無い場合

仮想アドレス

02	123

主記憶

ページ枠	ページ
0	01
1	03
2	07
3	06

2次記憶

ページ
00
01
02
03
04
05
07

ページ	ページ枠	フラグ		
		V	P	C
		0	100	0
01		1	110	1
02		0	110	0
03	1	1	111	1

主記憶上に無し

ページ枠に空き無し

ページフォルト発生

ページングの動作

主記憶上に無い場合

仮想アドレス

02	123

主記憶

ページ枠	ページ
0	01
1	03
2	07
3	06

2次記憶

ページ
00
01
02
03
06
07

ページ	ページ枠	フラグ		
		V	P	C
00		0	100	0
01	0	1	110	1
02		0	110	0
03		0	111	0

ページ枠を空けるために
03 をページアウト

ページングの動作

主記憶上に無い場合

仮想アドレス

02	123

実アドレス

1	123

主記憶

ページ枠	ページ
0	01
1	02
2	07
3	06

2次記憶

ページ
00
01
02
03
04
05
06
07

2次記憶から
ページイン

ページ	ページ枠	フラグ		
		V	P	C
02	1	1	110	0
03		0	111	0

主記憶上
の位置

主記憶上
に有り

ページングの動作

主記憶上に無い場合

仮想アドレス

03	999

主記憶

ページ枠	ページ
0	01
1	02
2	07
3	

2次記憶

ページ
00
01
02
04
05
06

ページフォルト発生

ページ	ページ枠	フラグ		
		V	P	C
00		0	100	0
01	0	1	110	1
02	1	1	110	0
03		0	111	0

03 はページアウトしたばかり

前回のページアウトが
01,06,07 のどれかであれば
ページフォルトは起きなかった

ページアウトするページ

ページアウトしたページを再度参照すると
ページフォルトが起こる

ページアウトするページは
近い将来に参照されないページがいい

どのページが「近い将来に参照されない」?

置き換えアルゴリズム

- OPT(optimal)
  - 今後最も長い期間使用されないページを選択
- FIFO(first in first out)
  - 最も早く主記憶に読み込んだページを選択
- LRU(least recently used)
  - 最も長い期間使用されていないページを選択
- LFU(least frequently used)
  - 最も参照回数の少ないページを選択

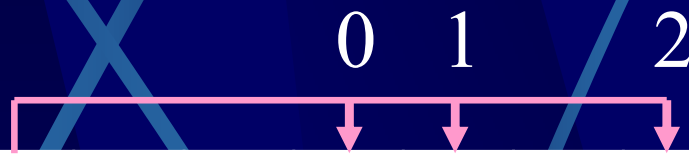
OPT(optimal)

- OPT(optimal)
  - 今後最も長い期間使用されないページを選択

主記憶

ページ枠	ページ	読み込み	前回使用	参照回数	次回使用
0	01	10回前	5回前	4回	2回後
1	03	7回前	7回前	1回	5回後
2	07	4回前	3回前	2回	7回後
3	06	2回前	1回前	2回	1回後

OPT



参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	0							
		1	1	1	1							
			2	②	4							
ページフォルト	p	p	p		p							

OPT

4 0 1

参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	0	0						
		1	1	1	1	3						
			2	2	4	4						
ページフォルト	p	p	p		p	p						

OPT

参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	0	0	0	0	1	1	1	1
		1	1	1	1	3	3	3	3	3	2	2
			2	2	4	4	4	4	4	4	4	4
ページフォルト	p	p	p		p	p			p		p	

ページフォルト 7 回

OPTの長所と短所

- OPTの長所
  - 最適なアルゴリズム: ページフォルト率が最低
- OPTの短所
  - 将来のページ参照が分かる必要あり

実用性は無し
= OPTを採用しているOSは存在しない
(他のアルゴリズムとの比較用)

参照の局所性
(locality of reference)

- 参照の局所性(locality of reference)
  - 主記憶へのアクセスは一部のアドレスに集中する可能性が高い
- 時間局所性
  - 最近参照されたページは近い将来に再度参照される可能性が高い
- 空間局所性
  - あるページが参照されると近くのページも近い将来に参照される可能性が高い

参照の局所性

本Aを
貸してください

本Aを
貸してください

本Bの第1巻を
貸してください

本Bの第2巻を
貸してください

本Bの第3巻を
貸してください

時間局所性

- 時間局所性
  - 最近参照されたページは近い将来に再度参照される可能性が高い

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

for ループ内では
変数 *i*, *sum* が繰り返し参照される

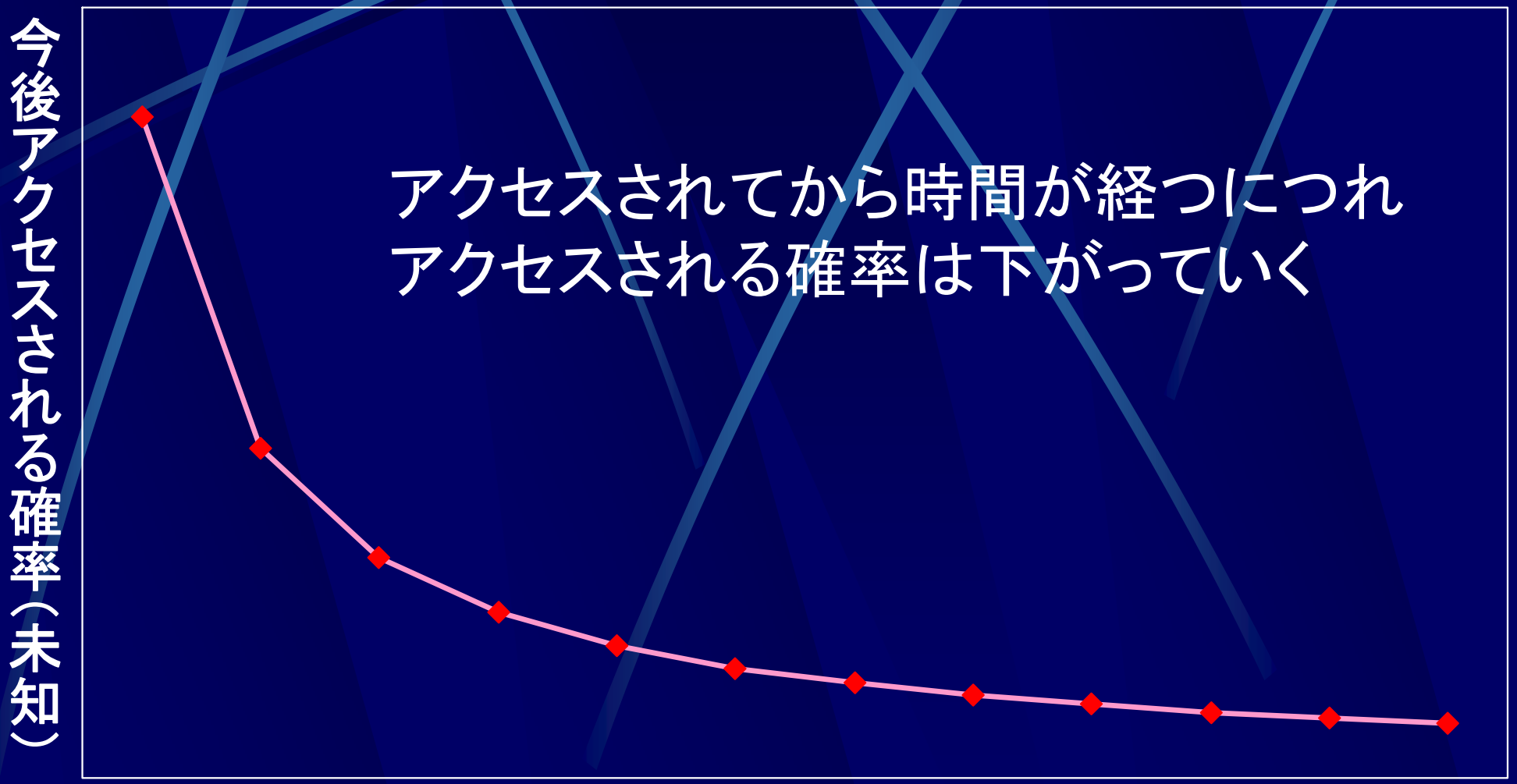
空間局所性

- 空間局所性
  - あるページが参照されると近くのページも近い将来に参照される可能性が高い

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

for ループ内では
$a[0], a[1], \ldots, a[n]$ が順に参照される

時間局所性



過去のアクセス間隔とアクセス確率の関係

局所性を利用した置き換え

多くのプログラムには時間局所性がある

- 最近参照されたページは近い将来に再度参照される可能性が高い

- 最近参照されていないページは近い将来に再度参照される可能性は低い

あまり参照されていないページをページアウトする

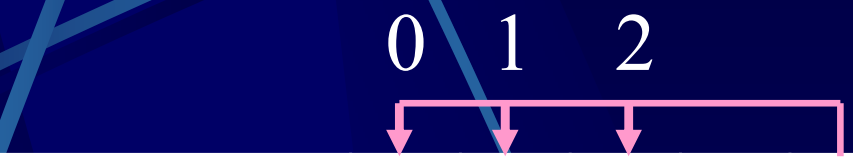
FIFO(first in first out)

- FIFO(first in first out)
  - 最も早く主記憶に読み込んだページを選択

主記憶

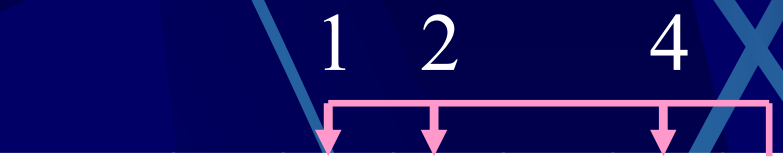
ページ枠	ページ	読み込み	前回使用	参照回数	次回使用
0	01	10回前	5回前	4回	2回後
1	03	7回前	7回前	1回	5回後
2	07	4回前	3回前	2回	7回後
3	06	2回前	1回前	2回	1回後

FIFO



参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0 1	0 1 2	0 1 2	4 1 2							
ページフォルト	p	p	p		p							

FIFO



参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	4	4						
		1	1	1	1	3						
			2	2	2	2						
ページフォルト	p	p	p		p	p						

FIFO

参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	4	4	4	4	1	1	1	1
		1	1	1	1	3	3	3	3	4	4	4
			2	2	2	2	2	0	0	0	2	2
ページフォルト	p	p	p		p	p		p	p	p	p	

ページフォルト 9 回

OPT ではページフォルト 7 回

キューによるFIFOの実装

- FIFOの実装
  - キューでページを管理する

参照ページ	0	1	2	0	4
ページ枠	0	0	0	0	1
(FIFOキュー)		1	1	1	2
			2	2	4
ページフォルト	p	p	p		p

1. 1番上のページを消す
2. ページを上にシフト
3. 1番下にページを加える

キューによるFIFOの実装

参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0 1	0 1 2	0 1 2	1 2 4	2 4 3	2 4 3	4 3 0	3 0 1	0 1 4	1 4 2	1 4 2
ページフォルト	p	p	p		p	p		p	p	p	p	

FIFOの長所と短所

- FIFOの長所
  - 実装が簡単
- FIFOの短所
  - 頻繁に使用するページでもページアウトされる
  - Belady の異常(Belady's anomaly)が起こる

FIFOの短所

ページ 0 は頻繁にアクセス

参照ページ	0	1	0	0	2	0	0	3
ページ枠	0	0 1	0 1	0 1	0 1 2	0 1 2	~~0~~ 1 2	1 2 3
ページフォルト	p	p			p			p

頻繁にアクセスされるページが
ページアウトしてしまう

Beladyの異常
(Belady's anomaly)

通常は

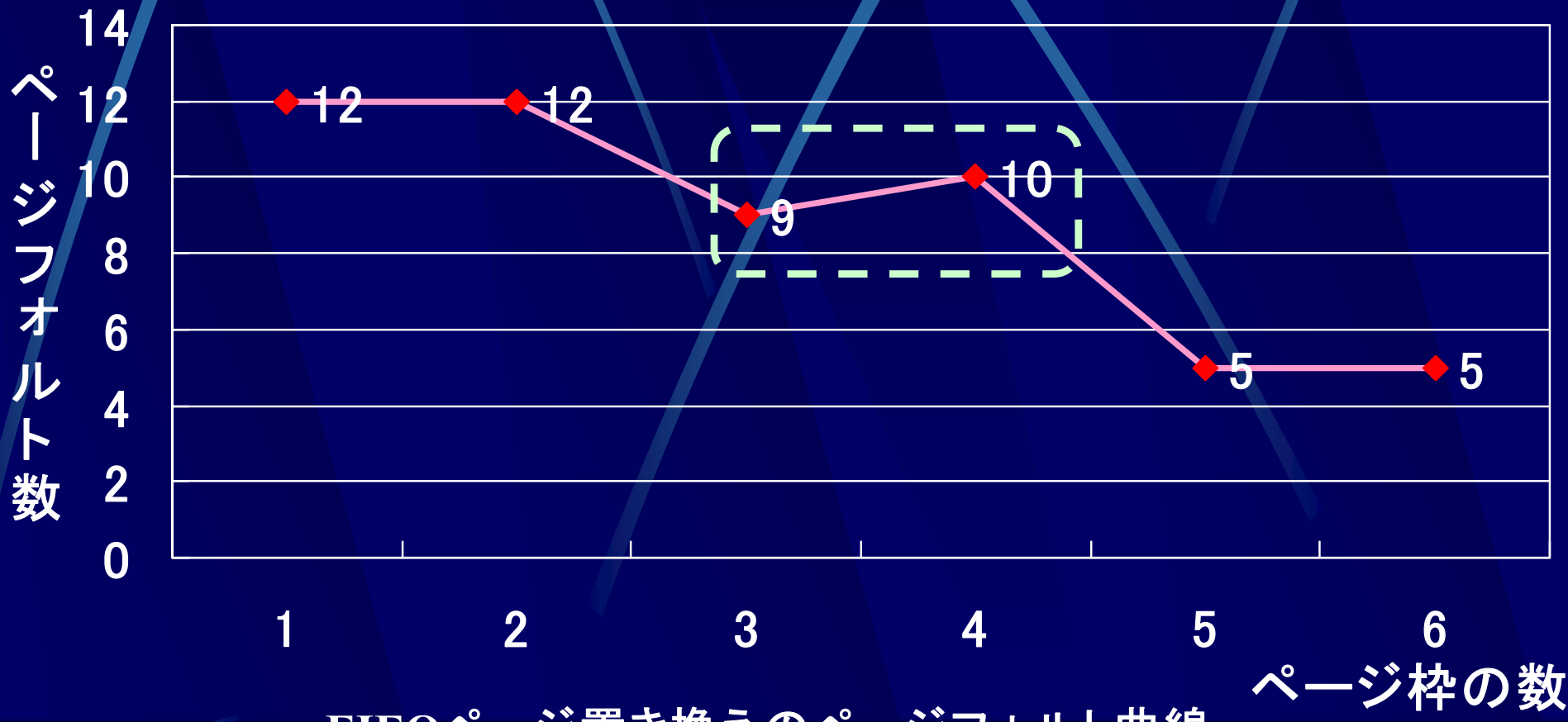
- Beladyの異常(Belady's anomaly)
  - FIFOでは、ページ枠数が増加したのにページフォルト数が増加してしまう場合がある

Beladyの異常

参照ページ		0	1	2	3	0	1	4	0	1	2	3	4
枠数3	ページ枠	0	0	0	1	2	3	0	0	0	1	4	4
			1	1	2	3	0	1	1	1	4	2	2
	ページフォルト 9 回			2	3	0	1	4	4	4	2	3	3
	ページフォルト	p	p	p	p	p	p	p			p	p	
枠数4	ページ枠	0	0	0	0	0	0	1	2	3	4	0	1
			1	1	1	1	1	2	3	4	0	1	2
				2	2	2	2	3	4	0	1	2	3
	ページフォルト 10 回				3	3	3	4	0	1	2	3	4
	ページフォルト	p	p	p	p			p	p	p	p	p	p

Beladyの異常

参照ページ	0	1	2	3	0	1	4	0	1	2	3	4



FIFOページ置き換えのページフォルト曲線

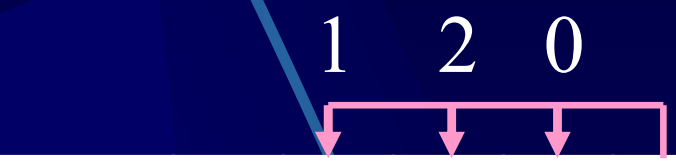
LRU(least recently used)

- LRU(least recently used)
  - 最も長い期間使用されていないページを選択

主記憶

ページ枠	ページ	読み込み	前回使用	参照回数	次回使用
0	01	10回前	5回前	4回	2回後
1	03	7回前	7回前	1回	5回後
2	07	4回前	3回前	2回	7回後
3	06	2回前	1回前	2回	1回後

LRU



参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	0							
		1	1	1	4							
			2	2	2							
ページフォルト	p	p	p		p							

LRU

2 0 4

参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	0	0						
		1	1	1	4	4						
			2	2	②2	3						
ページフォルト	p	p	p		p	p						

LRU

参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	0	0	0	0	0	0	2	2
		1	1	1	4	4	4	4	4	4	4	4
			2	2	2	3	3	3	1	1	1	1
ページフォルト	p	p	p		p	p			p		p	

ページフォルト 7 回
OPT ではページフォルト 7 回

LRUの長所と短所

- LRUの長所
  - 頻繁にアクセスするページはページアウトされない
  - Belady の異常が起こらない
- LRUの短所
  - 各ページの参照時刻の記録が必要

  ⇒ カウンタ, スタック, 参照ビット等のハードウェア支援が必要

LRUの実装

- カウンタによる実装
  - 各ページへのアクセス時のカウンタ値を記録
  - 最小のカウンタ値のページをページアウト
- 参照ビットによる実装
  - 各ページへアクセス時に1にセット
  - 0 のページを優先的にページアウト
- スタックによる実装
  - 各ページへのアクセス時にスタックの先頭に移動
  - スタックの末尾のページをページアウト

カウンタによるLRUの実装

- カウンタによる実装
  - ページへアクセスするときカウンタを増加
  - アクセスしたページにカウンタ値を記録

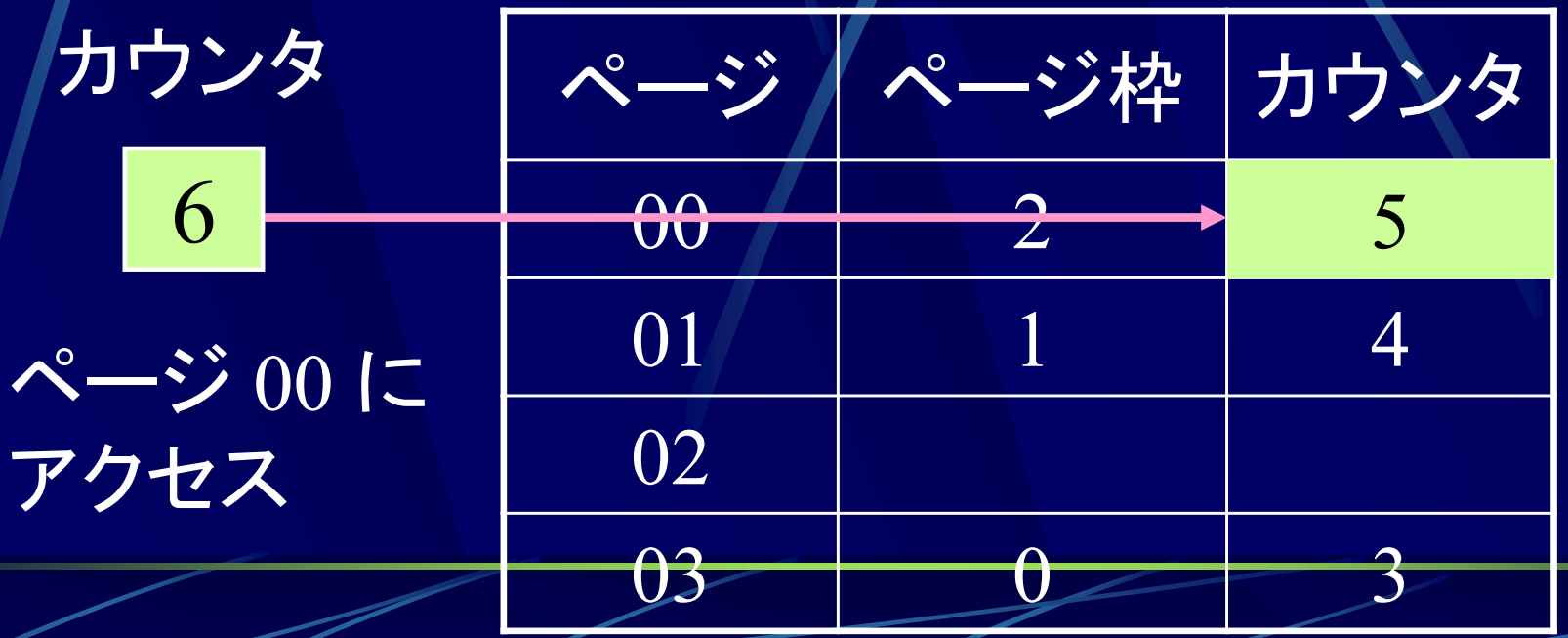
カウンタ

5

ページ	ページ枠	カウンタ
00	2	2
01	1	4
02		
03	0	3

カウンタによるLRUの実装

- カウンタによる実装
  - ページへアクセスするときカウンタを増加
  - アクセスしたページにカウンタ値を記録



カウンタによるLRUの実装

- カウンタによる実装
  - ページへアクセスするときカウンタを増加
  - アクセスしたページにカウンタ値を記録

カウンタ

6

ページ 02 にアクセス

ページ	ページ枠	カウンタ
		5
		4
02		
03	0	3

カウンタ値が最小のページをページアウト

カウンタによる実装

- カウンタによる実装
  - ページへアクセスするときカウンタを増加
  - アクセスしたページにカウンタ値を記録

カウンタ

7

ページ 02 にアクセス

ページ	ページ枠	カウンタ
00	2	2
01	1	4
02	0	6
03		

参照ビットによるLRUの実装

- 参照ビットによる実装
  - ページへアクセスするとき 1 にセット
  - 参照ビットが 0 のページを優先的にページアウト
  - 必要に応じて全ページの参照ビットを 0 にリセット

ページ	ページ枠	参照ビット
00	2	0
01	1	0
02		
03	0	0

参照ビットによるLRUの実装

- 参照ビットによる実装
  - ページへアクセスするとき 1 にセット
  - 参照ビットが 0 のページを優先的にページアウト
  - 必要に応じて全ページの参照ビットを 0 にリセット

ページ 00 にアクセス

ページ	ページ枠	参照ビット
00	2	1
01	1	0
02		
03	0	0

参照ビットによるLRUの実装

- 参照ビットによる実装
  - ページへアクセスするとき 1 にセット
  - 参照ビットが 0 のページを優先的にページアウト
  - 必要に応じて全ページの参照ビットを 0 にリセット

参照ビットが 0 のページをページアウト

ペー		参照ビット
00		1
01	1	0
02		
03	0	0

ページ 02 にアクセス

参照ビットによるLRUの実装

- 参照ビットによる実装
  - ページへアクセスするとき 1 にセット
  - 参照ビットが 0 のページを優先的にページアウト
  - 必要に応じて全ページの参照ビットを 0 にリセット

ページ 02 にアクセス

ページ	ページ枠	参照ビット
00	2	1
01		
02	1	1
03	0	0

スタックによるLRUの実装

- スタックによる実装
  - スタックでページを管理する

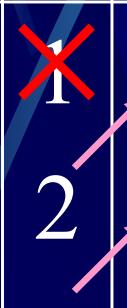
参照ページ	0	1	2	0	4
ページ枠 (FIFOキュー)	0	0 1	0 1 2	1 2 0	
ページフォルト	p	p	p		

参照したページを
一番下に移動

スタックによるLRUの実装

- スタックによる実装
  - スタックでページを管理する

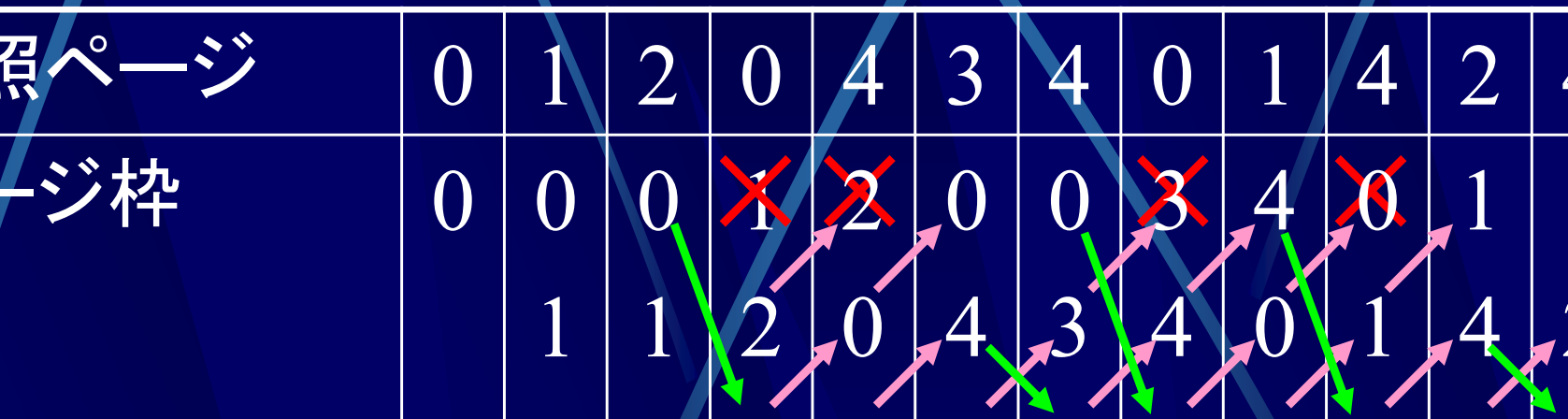
参照ページ	0	1	2	0	4
ページ枠 (FIFOキュー)	0	0	0	1	2
		1	1	2	0
			2	0	4
ページフォルト	p	p	p		p



1. 1番上のページを消す
2. ページを上にシフト
3. 1番下にページを加える

スタックによるLRUの実装

参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	1	2	0	0	3	4	0	1	1
		1	1	2	0	4	3	4	0	1	4	2
			2	0	4	3	4	0	1	4	2	4
ページフォルト	p	p	p		p	p			p		p	



LRUの実装

実装方法	長所	短所
カウンタ	LRUを実現	ハートウェアが必要 参照に時間が掛かる
参照ビット	ハードウェアが不要	LRUの近似
スタック	LRUを実現	ハードウェアが必要

LFU(least frequently used)

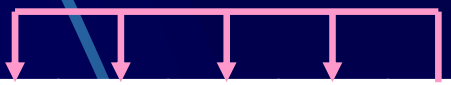
- LFU(least frequently used)
  - 最も参照回数の少ないページを選択

主記憶

ページ枠	ページ	読み込み	前回使用	参照回数	次回使用
0	01	10回前	5回前	4回	2回後
1	03	7回前	7回前	1回	5回後
2	07	4回前	3回前	2回	7回後
3	06	2回前	1回前	2回	1回後

0：2回
1：1回
2：1回

LFU



参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	0							
		1	1	1	4							
			2	2	2							
ページフォルト	p	p	p		p							

0：2回
4：1回
2：1回

LFU

参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	0	0						
		1	1	1	4	4						
			2	2	(2)	3						
ページフォルト	p	p	p		p	p						

LFU

参照ページ	0	1	2	0	4	3	4	0	1	4	2	4
ページ枠	0	0	0	0	0	0	0	0	0	0	0	0
		1	1	1	4	4	4	4	4	4	4	4
			2	2	2	3	3	3	1	1	2	2
ページフォルト	p	p	p		p	p			p		p	

ページフォルト 7 回
OPT ではページフォルト 7 回

LFUの長所と短所

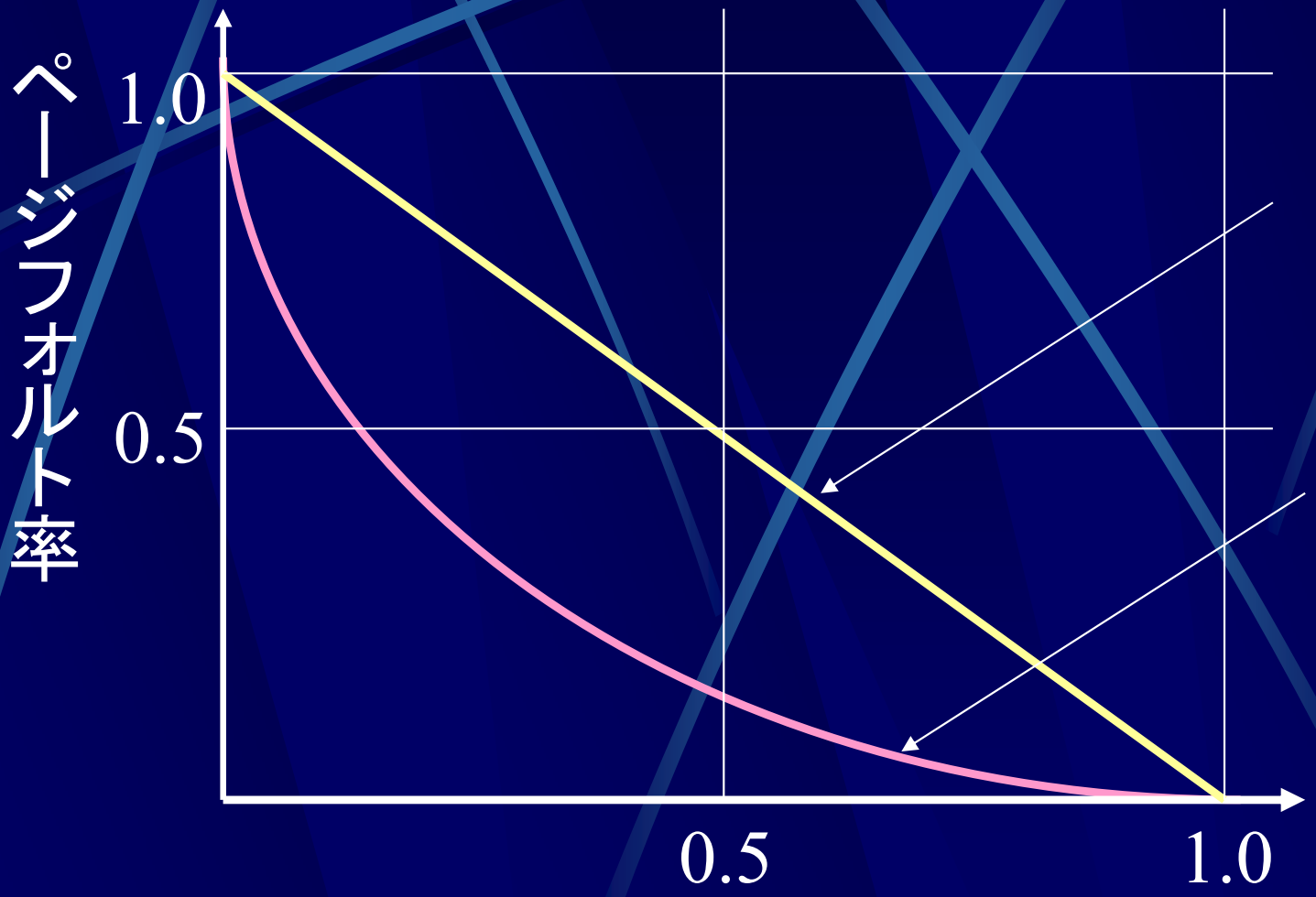
- LFUの長所
  - 頻繁にアクセスするページはページアウトされない
  - Belady の異常が起こらない
- LFUの短所
  - 参照に時間がかかる
  - 各ページの参照回数の記録が必要

  ⇒ カウンタ等のハードウェア支援が必要

置き換え技法の長所と短所

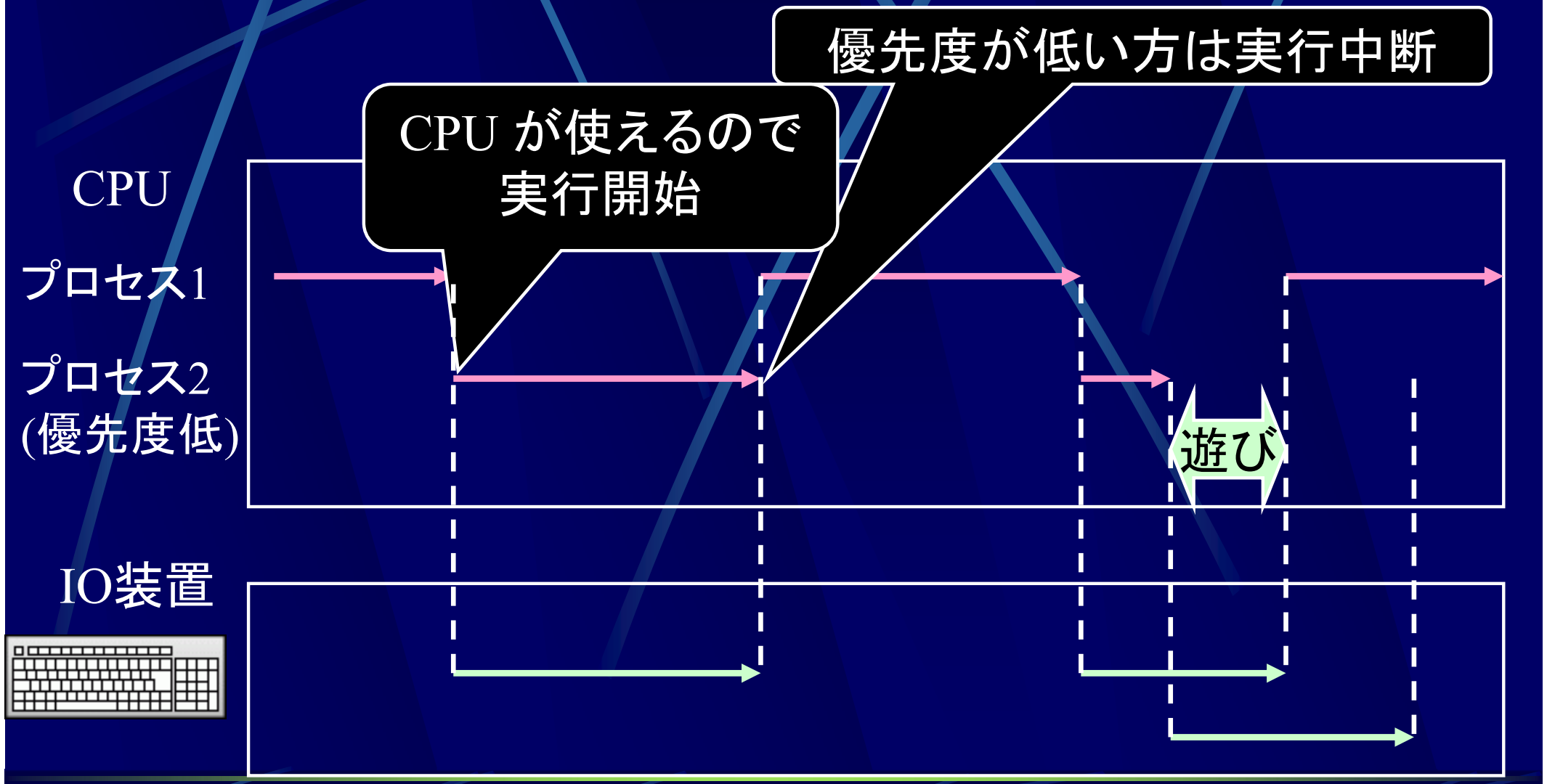
技法	長所	短所
OPT	最適	未来の参照が分からなければならない
FIFO	実装が簡単	頻繁に参照されるページがページアウト
LRU	参照の少ないページがページアウト	ハードウェアが必要
LFU	参照の少ないページがページアウト	ハードウェアが必要

ページフォルト曲線



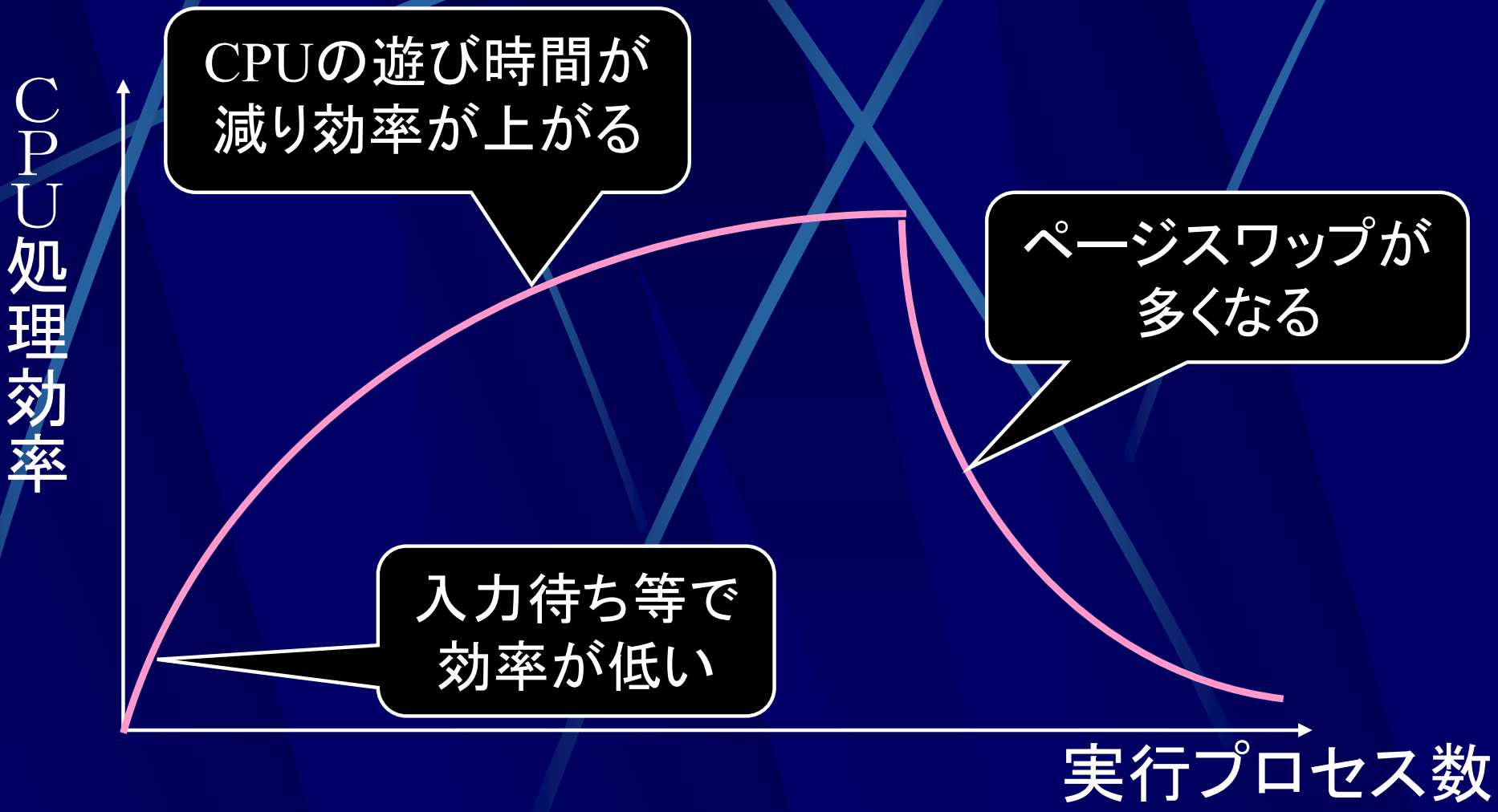
0.5を境にページフォルト率は急激に上昇

マルチプロセスの実行中の動作



マルチプロセスにすれば CPU の遊び時間を減らせる

実行プロセス数と処理効率



実行プロセスが増え過ぎると効率が下がる

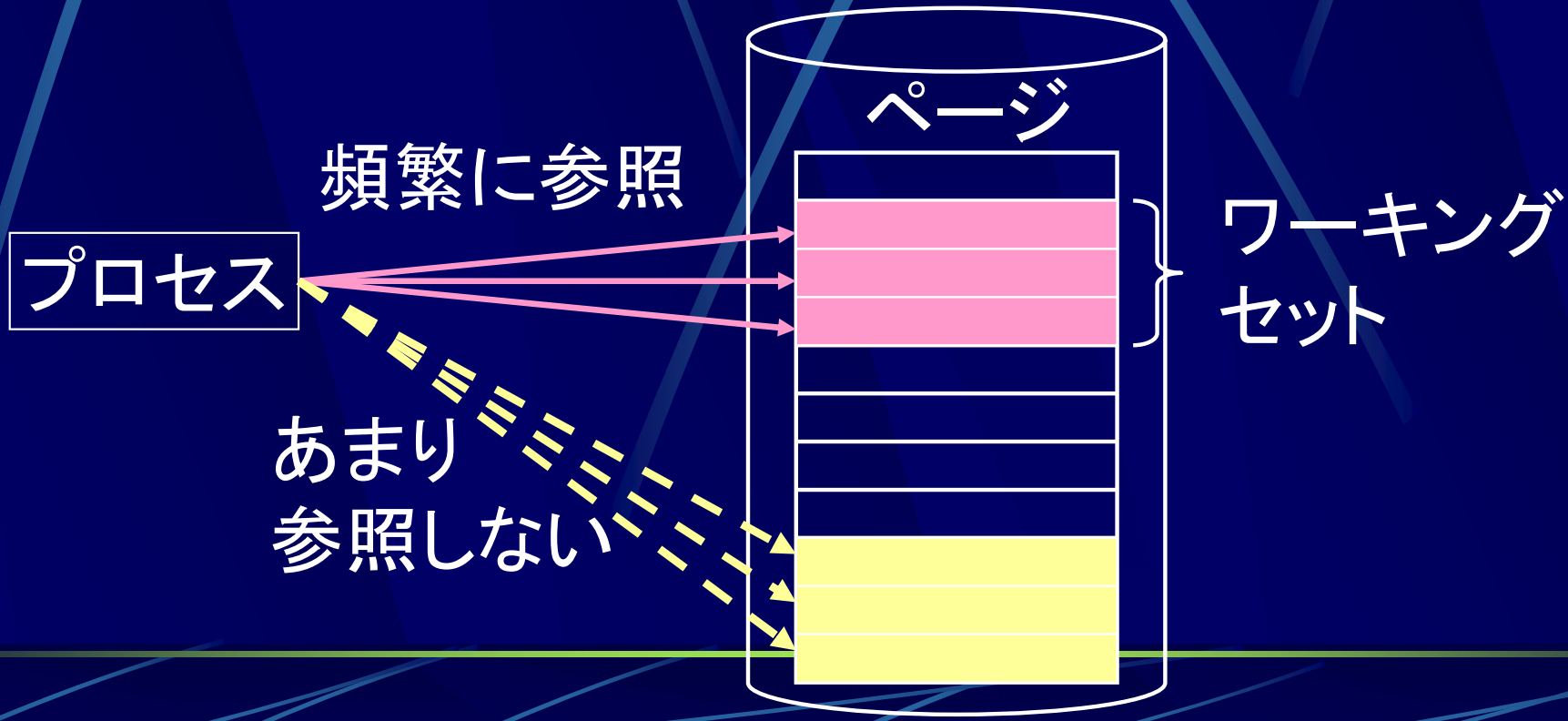
スラッシング(thrashing)

スラッシング(thrashing)

- スラッシング(thrashing)
  - 主記憶の容量が充分に無いため2次記憶への参照が繰り返し行われる状態
- スラッシングの原因
  - 非常に多くのプロセスが並行動作
  - 非常に大きな記憶領域を必要とするプロセスが動作

ワーキングセット(working set)

- ワーキングセット(working set)
  - プロセスが活発に参照するページの集合



ワーキングセットとページ枠

ワーキングセット＞ページ枠

頻繁にページフォルトが発生

スラッシング

各プロセスにワーキングセット以上の
ページ枠を割り当てる

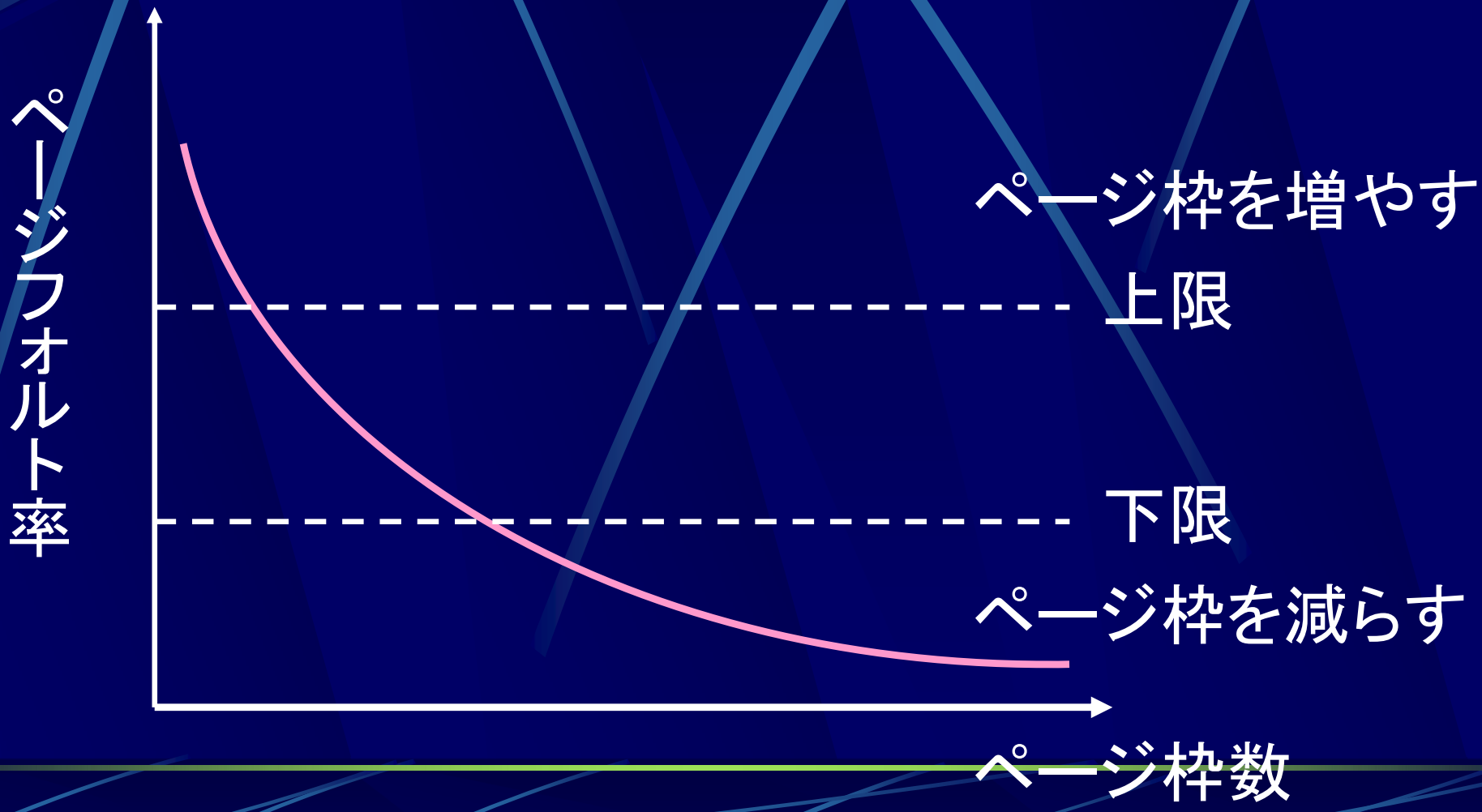
動的ページ置き換え

- 動的ページ置き換え
  - 各プロセスに割り当てるページ枠のサイズを動的に変える

ページフォルト発生頻度：大 ⇒ ページ枠増加
ページフォルト発生頻度：並 ⇒ ページ枠変更無し
ページフォルト発生頻度：小 ⇒ ページ枠減少

全てのプロセスでページフォルトが多発する場合は
優先度の低いプロセスを停止する

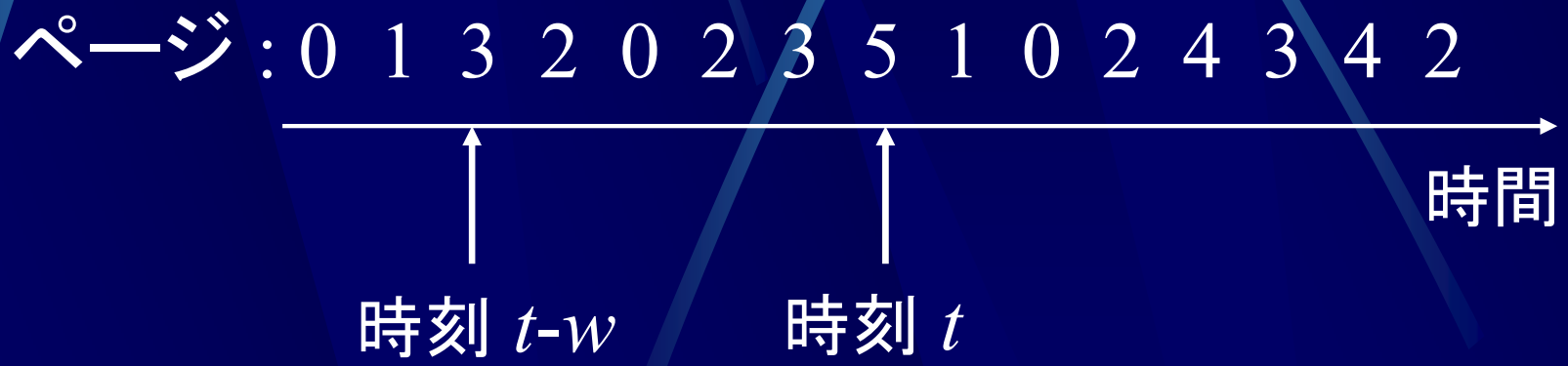
動的ページ置換え



ワーキングセットによる動的ページ置換え

- ワーキングセット

= 最近の w 時間以内にアクセスされたページ



時刻 t のワーキングセット : 3 2 0 5

w : ウィンドウサイズ

ワーキングセットによる動的ページ置換え

ウィンドウサイズ $w = 3$

参照ページ	0	1	2	1	4	3	4	0	1	4	2	4
ページ枠	0	0 1	0 1 2									
ページフォルト	p	p	p									

ワーキングセットによる動的ページ置換え

w 時間アクセスされなかったページは消去

ウィンドウサイズ $w = 3$

参照ページ	0	1	2	1	4	3	4	0	1	4	2	4
ページ枠	0	0 1	0 1 2	0 1 2								
ページフォルト	p	p	p									

ページ枠を2に減少

ワーキングセットによる動的ページ置換え

ウィンドウサイズ $w = 3$

参照ページ	0	1	2	1	4	3	4	0	1	4	2	4
ページ枠	0	0 1	0 1 2	0 1 2	2 1 4							
ページフォルト	p	p	p		p							

ページ枠を3に増加

ワーキングセットによる動的ページ置換え

ウィンドウサイズ $w = 3$

参照ページ	0	1	2	1	4	3	4	0	1	4	2	4
ページ枠	0	0 1	0 1 2	0 1 2	2 1 4	2 1 3	2 1 3	3 4 0	4 0 1	0 1 4	1 4 2	1 4 2
ページフォルト	p	p	p		p	p		p	p	p	p	

ページ枠を2に減少

ページ枠を3に増加

まとめ

- 置換え技法
  - 主記憶上のデータのうち、どれを二次記憶に追い出すか決定する
    - 今後使わないデータを追い出す

参照の局所性を利用して
今後使わないデータを推定

置き換えアルゴリズム

- OPT(optimal)
  - 今後最も長い期間使用されないページを選択
- FIFO(first in first out)
  - 最も早く主記憶に読み込んだページを選択
- LRU(least recently used)
  - 最も長い期間使用されていないページを選択
- LFU(least frequently used)
  - 最も参照回数の少ないページを選択

置き換え技法の長所と短所

技法	長所	短所
OPT	最適	未来の参照が分からなければならない
FIFO	実装が簡単	頻繁に参照されるページがページアウト
LRU	参照の少ないページがページアウト	ハードウェアが必要
LFU	参照の少ないページがページアウト	ハードウェアが必要

動的ページ置換え