東邦大学

生命（いのち）の科学で未来をつなぐ

ページ置き換えの動作とアルゴリズム スラッシング、LRUの実現法 全体のまとめ

東邦大学

前回までの議論

- メインメモリ（物理メモリ）が満杯になると
  入りきらないので
  ページを追い出さなければならない
- 参照列を使って、アルゴリズムを比較できる

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5

ＣＰＵがアクセスするページの番号の列

東邦大学

ページ置換えアルゴリズムは

- ページの追出し方で、
  次に欲しいページが主記憶上にあるか無いか
  （＝ページフォルトの発生率）が変わる
- 将来使うページが分かれば、それを
  なるべく残せばよい（OPTアルゴリズム）
  ⇒　未来のことは分からないので、
  現実には実現できない
- OPTに近いものとして、LRU が使われている
  - 後述のように擬似 LRU であるクロックアルゴリズムが広く使われている

ページ置換えアルゴリズムは

- 一般に物理メモリが小さい（不足がちだ）と、
  置換えの選択方法の影響が大

ページ置換えアルゴリズムは

- 一般に物理メモリが小さい（不足がちだ）と、
  置換えの選択方法の影響が大
- 物理メモリが本格的に不足すると、
  入れたものをすぐ追い出す「スラッシング」の
  状態になる

スラッシングとは

- ページフォルトが非常に頻繁に発生し、処理が進まなくなる状態のこと　（thrashing）

スラッシングとは

- ページフォルトが非常に頻繁に発生し、処理が進まなくなる状態のこと　（thrashing）
- 物理ページが必要量に比べて非常に少ないと発生
- 最近追出したページがまたすぐ必要となる状況
  - ページフォルトが非常に増える
    （実際、ある閾値を越えると急に増える傾向がある）
  - その結果、メモリの実効アクセス時間が非常に遅くなり処理が進まなくなる（システムの実行効率が急に減少）

スラッシングの回避方法

- ワーキングセット（欲しいメモリ量）が物理メモリに入る程度にする

スラッシングの回避方法

- ワーキングセット（欲しいメモリ量）が物理メモリに入る程度にする

  ①物理メモリを大きくする（増設する）

  - 物理メモリが大きくなれば、ページアウトが減る
  - ハードウェア投資必要、かつ、すぐには出来ない

スラッシングの回避方法

- ワーキングセット（欲しいメモリ量）が物理メモリに入る程度にする

  ①物理メモリを大きくする（増設する）

  - 物理メモリが大きくなれば、ページアウトが減る
  - ハードウェア投資必要、かつ、すぐには出来ない

  ②実行可能状態のプロセス数を減らす
  （プロセスの多重度を下げる）

  - メモリ上に同時に置かれるプロセス数を減らして欲しいメモリの量を減らす

スラッシングの回避方法

- ワーキングセット（欲しいメモリ量）が物理メモリに入る程度にする

  ①物理メモリを大きくする（増設する）

  - 物理メモリが大きくなれば、ページアウトが減る
  - ハードウェア投資必要、かつ、すぐには出来ない

  ②実行可能状態のプロセス数を減らす（プロセスの多重度を下げる）

  - メモリ上に同時に置かれるプロセス数を減らして欲しいメモリの量を減らす
  - 実行可能プロセスを一時的に待ちにする（プロセススワッピング・スワップアウト）

LRUの実現とクロックアルゴリズム

- LRUを忠実に実現するのは結構大変
  - 過去のアクセス時刻を覚えておかなければならない

LRUの実現とクロックアルゴリズム

- LRUを忠実に実現するのは結構大変
  - 過去のアクセス時刻を覚えておかなければならない
- 近似法　～　クロックアルゴリズムが使われる

LRUの実現とクロックアルゴリズム

- LRUを忠実に実現するのは結構大変
  - 過去のアクセス時刻を覚えておかなければならない
- 近似法　～　クロックアルゴリズムが使われる
  - ページテーブルにアクセスビットを付ける
    - ページが参照されると（ハードで）アクセスビットに１を上書き

東邦大学

LRUの実現とクロックアルゴリズム

- LRUを忠実に実現するのは結構大変
  - 過去のアクセス時刻を覚えておかなければならない
- 近似法　～　クロックアルゴリズムが使われる
  - ページテーブルにアクセスビットを付ける
    - ページが参照されると（ハードで）アクセスビットに１を上書き
  - （ソフトで）一定間隔でチェックする

LRUの実現とクロックアルゴリズム

- LRUを忠実に実現するのは結構大変
  - 過去のアクセス時刻を覚えておかなければならない
- 近似法　～　クロックアルゴリズムが使われる
  - ページテーブルにアクセスビットを付ける
    - ページが参照されると（ハードで）アクセスビットに１を上書き
  - （ソフトで）一定間隔でチェックする
  - 一定間隔内に参照があると１になっているはず

東邦大学

LRUの実現とクロックアルゴリズム

- LRUを忠実に実現するのは結構大変
  - 過去のアクセス時刻を覚えておかなければならない
- 近似法　～　クロックアルゴリズムが使われる
  - ページテーブルにアクセスビットを付ける
    - ページが参照されると（ハードで）アクセスビットに１を上書き
  - （ソフトで）一定間隔でチェックする
  - 一定間隔内に参照があると１になっているはず
    ⇒　１になっているページはそのまま残す
    ０のままのページを追出しの候補にする

LRUの実現とクロックアルゴリズム

- LRUを忠実に実現するのは結構大変
  - 過去のアクセス時刻を覚えておかなければならない
- 近似法　～　クロックアルゴリズムが使われる
  - ページテーブルにアクセスビットを付ける
    - ページが参照されると（ハードで）アクセスビットに１を上書き
  - （ソフトで）一定間隔でチェックする
  - 一定間隔内に参照があると１になっているはず
    ⇒　１になっているページはそのまま残す
    ０のままのページを追出しの候補にする
  - チェック後、クリア（0にする）

東邦大学

アクセスビットのついでにダーティビット

- ページテーブルにアクセスビット+ダーティビット

アクセスビットのついでにダーティビット

- ページテーブルにアクセスビット+ダーティビット
- アクセスビット
  - アクセスがあったら（ハードで）１にする
  - 一定間隔内にアクセスがあると１になっているはず
    ⇒　１になっているページはそのまま残す
    　　０のままのページを追出しの対象にする

アクセスビットのついでにダーティビット

- ページテーブルにアクセスビット+ダーティビット
- アクセスビット
  - アクセスがあったら（ハードで）１にする
  - 一定間隔内にアクセスがあると１になっているはず
    ⇒　１になっているページはそのまま残す
    　　０のままのページを追出しの対象にする
- ダーティビット
  - 追出すときにＨＤに書き戻すか否かを決める

アクセスビットのついでにダーティビット

- ページテーブルにアクセスビット+ダーティビット
- アクセスビット
  - アクセスがあったら（ハードで）１にする
  - 一定間隔内にアクセスがあると１になっているはず
    ⇒　１になっているページはそのまま残す
    　　０のままのページを追出しの対象にする
- ダーティビット
  - 追出すときにＨＤに書き戻すか否かを決める
  - アクセスが書込みであったら（ハードで）１にする

アクセスビットのついでにダーティビット

- ページテーブルにアクセスビット+ダーティビット
- アクセスビット
  - アクセスがあったら（ハードで）１にする
  - 一定間隔内にアクセスがあると１になっているはず
    ⇒　１になっているページはそのまま残す
    　　０のままのページを追出しの対象にする
- ダーティビット
  - 追出すときにＨＤに書き戻すか否かを決める
  - アクセスが書込みであったら（ハードで）１にする
    ⇒　１なら書込みがあったので、ＨＤに書き戻す
    　　０なら書込みが無かったのでそのまま捨ててよい

メモリへの書き込み結果をＨＤに反映するため

ページアウトデーモン

- 空きページを裏で作っておくサーバープロセス

ページアウトデーモン

- 空きページを裏で作っておくサーバープロセス
- なぜ：

ページアウトデーモン

- 空きページを裏で作っておくサーバープロセス
- なぜ：
  - 空きページが欲しい時に追出すのでは、時間がかかり
    ページインがかなり待たされることになる

東邦大学

ページアウトデーモン

- 空きページを裏で作っておくサーバープロセス
- なぜ：
  - 空きページが欲しい時に追出すのでは、時間がかかり
    ページインがかなり待たされることになる
  - あらかじめ空きページを一定量作りためておく
    ⇒　作りためるプロセスを、裏で動かしておく
    プールが一定量を切ると、ページアウト動作をし
    一定量になるまで作りためる

ページアウトデーモン

- 空きページを裏で作っておくサーバープロセス
- なぜ：
  - 空きページが欲しい時に追出すのでは、時間がかかり
    ページインがかなり待たされることになる
  - あらかじめ空きページを一定量作りためておく
    ⇒　作りためるプロセスを、裏で動かしておく
    プールが一定量を切ると、ページアウト動作をし
    一定量になるまで作りためる
  - ページインで空きページが必要になったとき、
    プールしてある中から１つ取り出して使えばよい

全体のまとめ

- ページの追出し（ページアウト）は、
  物理ページに空きがなくなったときに
  「不要な」ページを選んで追出すこと
- 追出す（不要な）ページの選択アルゴリズムには
  たとえば FIFO や LRU や OPT などが考えられ、
  LRU が広く使われている
- LRU の（擬似的な）実現法としてクロックアルゴリズムが使われている
- 物理メモリが不足するとスラッシングが起こる
  回避にはメモリ増設・プロセススワッピングなど

東邦大学

ページ追出しの考え方が
理解できましたか？

次へ

東邦大学