4. DB Buffer Management

Theo Härder

Main reference:

DB Buffer Management -- Contents

- Goal: realization of efficient, page-based in-memory processing
  - maximum possible avoidance of physical I/O
  - replacement algorithms without and with context knowledge
- Role of DB buffer management
  - course of access to the DB buffer
  - comparison with similar functionality in operating systems (OS)
- Locality
  - various measures for locality
  - characterization by LRU-stack-depth distributions and reference density
- Memory allocation and search in the DB buffer
- Page replacement algorithms
  - classification of replacement algorithms
    - LRU, CLOCK, GCLOCK
    - LRD, LRU-K ...
- Replacement algorithms - utilization of context knowledge
- DB Buffer Management - further problems
Realization of DBS

Role of DB Buffer Management in a DBMS

On-demand replacement

Memory allocation & search

Page replacement algorithms

On-demand replacement

Context knowledge

Variable-length pages

Realization of DBS

Page Reference Strings (PRSs)

Every data request is a logical page reference

Task of DB buffer management: minimization of physical page references

PRS = <r1, r2, ..., rn> with ri = (Ti, Di, Pi)

- Ti accessing transaction
- Di referenced DB partition
- Pi referenced DB page

Determination of windows in PRS w.r.t. specific transactions, transaction types, and DB partitions are useful for the analysis of the reference behavior

How can we use reference string information for
- characterization of reference behavior?
- determination of locality and sequentiality?
- support of an effective page replacement?
Properties of DB Reference Strings

- Typical reference pattern in DBS

1. Sequential search
   - Example: scanning of entire record types (tables)

2. Hierarchical paths
   - Example: search using B*-trees

3. Cyclic paths
   - Example: processing of sets (1:n)-relationships, search in DBTT/data pages

Comparison with OS Functionality

- Replacement algorithms of DB buffer are implemented in SW – page replacement in address spaces of Virtual Storage is HW supported

- Page reference vs. addressing
  - after a Fix call, a DB page can be referenced several times until Unfix
  - different page reference behavior
  - other replacement algorithms?

- Can OS file buffers be used as DB buffers?
  1. Access to file buffer is expensive (SVC: supervisor call)
  2. DB-specific reference patterns cannot be exploited systematically
     for example, OS replacement algorithms are not aligned to cyclic sequential or arborescent access sequences
  3. Normal file systems do not offer a suitable interface for prefetching
     based on page contents or reference patterns, DBMS are enabled to predict the reference behavior (e.g., for table scans); prefetching may enormously enhance the performance
  4. Selective writes of pages at specific points in time
     (e.g., for logging) are not always possible in existing file systems
     - DBMS must implement own buffer management, 64-bit architectures make not a difference!
Sequentiality

- PRSs typically show phases of sequentiality and locality
- Sequential reference sequence (SRS):
  - two subsequent references \( r_i \) and \( r_{i+1} \) belong to a sequential reference sequence, if
  \[ P_{i+1} - P_i = 0 \text{ or } 1 \]
  - i.e., subsequent accesses reference adjacent DB pages
- **Algorithm**
  - the page reference string is completely scanned; alternatively, the sequence of arriving references can be analyzed
  - as long as the above condition is satisfied, all consecutive references belong to an SRS; otherwise, a new SRS starts
- Length of a sequential reference sequence (LRS):
  - LRS is the number of different pages referenced in an SRS
  - PRS-1: A A B D E F F H contains
    - (AABB) with LRS(1) = 2, (DEEF) with LRS(2) = 3
    - and (H) with LRS(3) = 1
- Measure for sequentiality:
  - cumulative distribution of SRS lengths LRS(i)
    - \( S(x) = \Pr(\text{SRS length} \leq x) \)
    - PRS-1 yields: \( S(1)=0.33, S(2)=0.67, S(3)=1.0 \)
- Sequentiality allows optimization by (asynchronous) prefetching of pages

Locality

- Enhanced reuse probability for pages just referenced
- **Fundamental prerequisite for**
  - effective DB buffer management (page replacement)
  - use of storage hierarchies
- How can locality be measured?
  - **Working-Set model**
    - reference string
      - \( A B A C A B A B D C E F G H \)
    - window size \( w \)
      - \( w=8 \)
    - working set size \( W \)
      - \( W(t_1, w=8) = 3 \)
      - \( W(t_2, w=8) = 8 \)
    - current locality at time \( t \):
      - \( AL(t, w) = \frac{W(t, w)}{W} \)
    - average locality:
      - \( L_w = \frac{1}{n} \sum_{t=1}^{n} AL(t, w) \)
Realization of DBS

Locality

Role of DB buffer mgmt

Sequentiality vs. Locality

W (t, w)

locality

window size w

L (w)

locality

window size w

S (x)

low sequentiality

high sequentiality

SRS length x

Relative Reference Matrix (DOA Load)

~ 17 500 transactions, 1 million page references on ~ 66 000 pages

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
<th>P11</th>
<th>P12</th>
<th>P13</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT1</td>
<td>9.1</td>
<td>3.5</td>
<td>3.3</td>
<td>5.0</td>
<td>0.9</td>
<td>0.4</td>
<td>0.1</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22.3</td>
</tr>
<tr>
<td>TT2</td>
<td>7.5</td>
<td>6.9</td>
<td>0.4</td>
<td>2.6</td>
<td>0.0</td>
<td>0.5</td>
<td>0.8</td>
<td>1.0</td>
<td>0.3</td>
<td>0.2</td>
<td>0.0</td>
<td></td>
<td></td>
<td>20.3</td>
</tr>
<tr>
<td>TT3</td>
<td>6.4</td>
<td>1.3</td>
<td>2.8</td>
<td>0.0</td>
<td>2.6</td>
<td>0.2</td>
<td>0.7</td>
<td>0.1</td>
<td>1.1</td>
<td>0.4</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td>15.6</td>
</tr>
<tr>
<td>TT4</td>
<td>0.0</td>
<td>3.4</td>
<td>0.3</td>
<td>6.8</td>
<td>0.6</td>
<td>0.4</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11.6</td>
</tr>
<tr>
<td>TT5</td>
<td>3.1</td>
<td>4.1</td>
<td>0.4</td>
<td>0.0</td>
<td>0.5</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.2</td>
</tr>
<tr>
<td>TT6</td>
<td>2.4</td>
<td>2.5</td>
<td>0.6</td>
<td>0.7</td>
<td>0.9</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7.4</td>
</tr>
<tr>
<td>TT7</td>
<td>1.3</td>
<td>2.6</td>
<td></td>
<td>2.3</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.2</td>
</tr>
<tr>
<td>TT8</td>
<td>0.3</td>
<td>2.3</td>
<td>0.2</td>
<td>0.0</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.9</td>
</tr>
<tr>
<td>TT9</td>
<td>0.0</td>
<td>1.4</td>
<td>0.0</td>
<td></td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.6</td>
</tr>
<tr>
<td>TT10</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
<td>1.0</td>
<td>0.1</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>TT11</td>
<td>0.9</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.1</td>
<td></td>
</tr>
<tr>
<td>TT12</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>30.3</td>
<td>26.6</td>
<td>11.0</td>
<td>9.4</td>
<td>8.3</td>
<td>4.9</td>
<td>4.1</td>
<td>3.3</td>
<td>1.4</td>
<td>1.6</td>
<td>0.6</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

partition size (%)  

31.3 6.3 8.3 17.8 1.0 20.8 2.6 7.3 2.6 1.3 0.8 0.0 0.0 100.0

% referenced  

11.1 16.6 8.0 2.5 18.1 1.5 9.5 4.4 5.2 2.7 0.2 13.5 5.0 6.9
How can locality be characterized?
- LRU-stack-depth distribution provides a measure for locality (more precise than Working-Set approach)
- LRU-stack contains all already referenced pages in the sequence of their "age to their most recent reference"

Determination of stack-depth distribution
- a counter is maintained per stack position
- re-reference of a page increases the counter for the resp. stack position

Counter values correspond to reuse frequency
when applying LRU page replacement, the hit rate (resp. page fault rate) for a specific buffer size can be directly determined from the stack-depth distribution
Realization of DBS

Locality

Role of DBS buffer management

Memory allocation & search

Page replacement algorithms

On-demand replacement

Context knowledge

Variable-length pages

Real LRU-stack-depth Distributions

- LRU-stack-depth distribution of Mix 50
  - string length: 99975 logical references
  - number of different pages in string: 5245

- LRU-stack-depth distribution of Mix 40
  - string length: 130366 logical references
  - number of different pages in string: 3553

Reference Density Curves

- relative frequency of page types in the example
  - data and index structures: 93.8%
  - address translation tables: 6.1%
  - free placement administration: 0.1%
Realization of DBS

Memory Allocation

in the DB Buffer

- global (a single shared buffer area)
- local/partitioned buffer areas

Static
- uniform partitions
- adjusted partitions

Dynamic

Choice of partitioning:
- own buffer area per transaction
- TA-type-related buffer areas
- page-type-related buffer areas
- DB-(partition)-specific buffer areas

Dynamic Buffer Allocation – Working-Set Approach

- Per buffer partition P, the Working Set (WS) should be kept in the buffer; pages not belonging to the Working Set can be replaced.
- When a page fault condition occurs, the Working Set must be known to determine the replacement candidate (victim).
  - window size per partition: \( w(P) \)
  - reference counter per partition: \( RC(P) \)
  - least reference time for page i: \( LRT(P, i) \)
  - replaceable are those pages, for which \( RC(P) - LRT(P, i) > w(P) \)
- Window size is critical parameter -> danger of thrashing

<table>
<thead>
<tr>
<th>Reference string for P1:</th>
<th>A</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>A</th>
<th>G</th>
<th>H</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference string for P2:</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>RC (P1) = 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( LRT(P1, H) = 8 \)
- \( LRT(P1, A) = 5 \)
- \( LRT(P1, C) = 3 \)
Search in the DB Buffer

- Sequential search of buffer frames
  - very high search overhead
  - danger of many paging exceptions in Virtual Storage systems

- Use of auxiliary structures (entry per buffer frame)
  1. unsorted or sorted table
  2. table with chained entries
    - lower update overhead
    - allocation in LRU sequence possible
  3. search trees (e.g., AVL trees, m-way trees)
    - best solution
  4. hash table with overflow chain
    - best solution

\[ h(P_i) = k \]

Page Replacement Algorithms

- **Classification**
  - **method classes**
  - **preplanning**
    - Program analysis, Pre-analysis of data demand
    - high fault rate, imprecise supersets
  - **prefetching**
    - physical data structuring, clustering, processing knowledge
    - data model related, (hierarchical), speculative decisions
  - **demand fetching**
    - no pre-actions
    - locality preservation in DB buffer

- **Basic assumption for replacement algorithms**
  - references
    - most recent past
    - next future
    - reference behavior similar
Reference Behavior and Replacement Algorithms

- **Reference behavior in DBS**
  - typically high locality: optimization through replacement algorithms
  - sometimes sequentiality or random workload (RANDOM references)
- **Fundamental relationships** determining the page fault rate

\[
\text{page fault rate} = \frac{D}{N} \times 100\% 
\]

- **Combinations**
  - references: random random locality locality
  - replacement: RANDOM OPT RANDOM OPT

⇒ Borderline cases of reference behavior and replacement algorithms reveal optimization opportunities

Dealing with Modified Pages in the DB buffer

- Replacement of a modified page requires its prior (synchronous) propagation into the DB
  ⇒ response time degradation
- **Dependency to the chosen propagation strategy:**
  - **Force:** all updates of a transaction have to be propagated not later than EOT into the DB („write-through“)
    + always unmodified pages available for replacement, in general
    + simplified recovery (after crash, all updates of committed transactions are already propagated into the DB)
    - high I/O overhead
    - large response time increase for writer transactions
  - **NoForce:** no propagation of updated upon EOT („deferred write-back“)
    + page can be modified several times before propagation applies
    (smaller I/O overhead, improved response times)
    + anticipatory (asynchronous) propagation of modified pages
    enables also in case of NoForce to predominantly replace unmodified pages

⇒ Synchronous DB writes can be avoided as far as possible
Realization of DBS

Role of DB buffer mgmt

Locality

Memory allocation & search

Page replacement algorithms

On-demand replacement

Context knowledge

Variable-length pages

Criteria for the Replacement of Buffer Pages

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT</td>
<td>age</td>
</tr>
<tr>
<td>RANDOM</td>
<td>reference</td>
</tr>
<tr>
<td>LFU</td>
<td>reference frequency</td>
</tr>
<tr>
<td>FIFO</td>
<td>other criteria</td>
</tr>
<tr>
<td>LRU</td>
<td>pre-knowledge</td>
</tr>
<tr>
<td>CLOCK</td>
<td>-</td>
</tr>
<tr>
<td>GCLOCK</td>
<td>-</td>
</tr>
<tr>
<td>LRD (V1)</td>
<td>-</td>
</tr>
<tr>
<td>LRD (V2)</td>
<td>-</td>
</tr>
<tr>
<td>LRU-K</td>
<td>-</td>
</tr>
</tbody>
</table>

Least Frequently Used and First-In First-Out

Algorithm LFU

- Reference counter per page is incremented upon each page reference
- Replacement of page having the lowest reference frequency

RC

2
4
1
3
3
6
1
3

Age of a page is not considered!
Least Frequently Used and First-In First-Out

- **Algorithm FIFO**
  - Oldest page in the DB buffer is replaced
  - References while staying the buffer are not considered

```
  A  B  C  D
  ↓   ↓   ↓   ↓
  E   F   G   H
```

→ Suitable only for strict sequential reference behavior

---

Least Recently Used (LRU)

- **Example (buffer size 4)**

Reference of page C

```
A —— C —— D
A —— B —— D
```

Reference of page E

```
A —— E —— B
A —— A —— C
```

- **Distinction between**
  - Least Recently **Referenced**
  - Least Recently **Unfixed**

```
Fix A
Fix B
Unfix B
Unfix A
```
Realization of DBS

Locality

Memory allocation & search

Page replacement algorithms

On-demand replacement

Context knowledge

Variable-length pages

Clock (Second Chance)

- **Algorithm**
  - Extension of FIFO
  - Reference bit per page, which is set (1) upon each reference to the page
  - Replacement only applies when bit is already reset (0); otherwise, the bit is reset

```
0
0 1
1
0 1
1
```

- approximate consideration of the most recent reference time

Page Replacement Algorithms - Example

<table>
<thead>
<tr>
<th>page reference sequence</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>5</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>3</th>
<th>2</th>
<th>5</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>LRU</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>FIFO</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>CLOCK</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

© 2011 AG DBIS
Realization of DBS

Locality

Role of DB buffer management

Memory allocation & search

Page replacement algorithms

On-demand replacement

Context knowledge

Variable-length pages

GCLOCK (Generalized CLOCK)

Algorithm
- Reference counter (RC, instead of reference bit) is maintained per page.
- Replacement only if a page has counter value 0.
- Otherwise, the counter is decremented and the next page is inspected.

<table>
<thead>
<tr>
<th>data</th>
<th>index</th>
<th>addr-tab.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Versions of weighted increments
V1: $RC(j) = F_i$
   $RC(j) = RC(j) + R_i$
V2: $RC(j) = F_i$
   $RC(j) = R_i$

Parameters of the algorithm
- Initial values for reference counters ($F_i$).
- Choice of the decrement value.
- Counter is incremented upon a re-reference ($R_i$).
- Use of page-type of page-specific weights.

Least Reference Density (LRD)

Algorithm
- If a page has to be replaced, reference density is determined for all pages in the DB buffer.
- Reference density = reference frequency for a given reference interval.
- Replacement candidate is page having lowest reference density.
- Variant 1: reference interval corresponds to the age of a page.

Computation of the reference density
- Global counter GC: total number of all references.
- Arrival time AT: GC value when the page enters the buffer.
- Reference counter RC.
- Reference density $RD(j) = \frac{RC(j)}{GC - AT(j)}$.
Least Reference Density (2)

- **Variant 2: constant interval size**
  - artificial aging of pages: older references get a smaller weights, whenever their reference density is determined
  - *periodical reduction* of reference counters to reduce the influence of older references
  - reduction of RC by division or subtraction:

\[
RC(i) = \frac{RC(i)}{K_1} \quad (K_1 > 1) \quad \text{or} \quad RC(i) = \begin{cases} 
RC(i) - K_2 & \text{if } RC(i) - K_2 \geq K_3 \\
K_3 & \text{otherwise} \quad (K_2 > 0, K_3 \geq 0)
\end{cases}
\]

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>t2</td>
<td>t3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC(A)</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC(B)</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC(C)</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC(D)</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC(E)</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC(F)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**LRU-K**

- **Recording of the K past reference times** (per page in the DB buffer)
  - More expensive recording, method does not need explicit “aging”
  - At inspection time t, the reference string r_1, r_2, ..., r_t be given.

**Backward K-Distance b_i(P, K)** is the distance measured in references backwards to the K-recent reference onto page P:

- \[ b_i(P, K) = x, \text{ if } r_{t-x} \text{ owns value } P \text{ and there exist exactly } K-1 \]
  - other values \( (t-x < i \leq t) \) with \( r_i \) owning \( P \)

- \[ b_i(P, K) = \infty, \text{ if } P \text{ is not referenced at least } K \text{ times in } r_1, r_2, ..., r_t \]

**Example (K=1, 2, 3)**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
</tr>
</tbody>
</table>

\[
b_{100}(A,K) = \begin{cases} 
K=1 & K=2 & K=3 & K=4 \\
b_{100}(B,K) = \\
b_{100}(C,K) = 
\end{cases}
\]
LRU-K (2)

- $b_t(P, K)$ of the buffer pages are needed to select a replacement candidate!
  - Special handling for pages with $b_t(P, K) = \infty$
  - How does LRU-K correspond to LRD? Approximation of reference density?

- LRU-2 (i.e., $K=2$) represents the best solution, in general 3
  - Results similarly good as for $K > 2$, however simpler realization
  - Faster reaction in case of reference variations than for larger $K$

---

Simulation of Page Replacement Algorithms

- Characteristics of DB, TA load and logical page reference strings

<table>
<thead>
<tr>
<th></th>
<th>PAGE 1</th>
<th>PAGE 2</th>
<th>PAGE 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td>Percentage of references to page-local data</td>
<td>PBS 20%</td>
<td>PBS 20%</td>
<td>PBS 20%</td>
</tr>
<tr>
<td>Percentage of references to non-page-local data</td>
<td>PBS 80%</td>
<td>PBS 80%</td>
<td>PBS 80%</td>
</tr>
<tr>
<td>Hits</td>
<td>69.6%</td>
<td>69.6%</td>
<td>69.6%</td>
</tr>
<tr>
<td>Misses</td>
<td>30.4%</td>
<td>30.4%</td>
<td>30.4%</td>
</tr>
<tr>
<td>Reference frequency distribution of SOAP access to the other pages</td>
<td>0.000, 0.000, 0.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of references to shared pages</td>
<td>1.105</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>Bold mark buffer space %</td>
<td>5.22</td>
<td>5.22</td>
<td></td>
</tr>
<tr>
<td>Number of deferred transactions</td>
<td>42</td>
<td>42</td>
<td></td>
</tr>
</tbody>
</table>

### Realization of DBS

#### Locality

**Role of DB buffer mgmt**

**Simulation Results**

**On-demand replacement**

**Memory allocation & search**

**Page replacement algorithms**

**Context knowledge**

**Variable-length pages**

### Replacement Algorithm – Use of Context Knowledge

**Problems for LRU-based methods**

- **T1**: long sequential scan with very fast page requests
  - effects on **T1**: pages of **T1** are replaced if it proceeds with "slow" requests – even showing high reference locality

  T1: A B C D E F G H

  T2: Z Z Z

  - cyclic referencing (*loop*) of a set of pages (# pages > # frames)

  LRU: A B C D E A B C D E

  MRU: A B C D

  ➔ internal thrashing

---

© 2011 AG DBIS
**Replacement Algorithm – Use of Context Knowledge (2)**

- Exploitation of context knowledge for set-oriented requests
  - improvements in relational DBS possible

- Query execution plans (QEPs) by query optimizer
  - access characteristics/set of referenced pages can be predicted or estimated when QEPs are constructed
  - access pattern always contains cycles/loops
    - (at least control page – data page: nested loop join, etc.)
  - cost estimate for QEPs can consider available frames
  - upon execution, min. # of frames for a QEP is communicated to buffer mgmt

- **Hot Set**: set of pages in reference cycle
  - typical characteristics of page fault rate (PFR) for specific operations

![Diagram of PFR with hot points]
**Realization of DBS**

- Role of DB buffer management
- Locality
- Memory allocation & search
- Page replacement algorithms
- On-demand replacement
- Context knowledge
- Variable-length pages

### Hot Set Model

- **Hot Point**: abrupt change in the PFR, e.g., caused by a loop in a join
- **Hot Set Size (HSS)**: largest Hot Point smaller than the available DB buffer
- Query optimizer computes HSS for various QEPs (estimation of #frames)

#### Example

```
SELECT * FROM DEPT X, EMP Y
WHERE X.DNO = Y.DNO
AND ...
```

- **Application characteristics**
  - consideration of HSS in the total costs
  - selection dependent on available DB buffer size
  - binding at runtime possible

---

### DB Buffer Management / Variable-Length Pages

- Allocation example with variable-size pages and fixed-size frames

```
......
```

- **Problems**
  - buffer fragmentation

```
......
```

- replacement of several pages to satisfy a new page request

```
......
```

- **Goals**
  - maximization of buffer use (memory optimization)
  - minimization of I/O through use of locality in the reference behavior

- Proposal for an algorithm: VAR-PAGE-LRU* - what does LRU here mean?

- Further task: Caching and speculative prefetching of documents in storage hierarchies (tertiary storage)**

Page Replacement in Virtual Memory

- **Page fault**
  - \( P_i \) in system buffer (SB) virtual, but not in SB real (main memory)

- **Database fault**
  - \( P_i \) not in SB virtual, page frame for \( P_i \) however in SB real

- **Double page fault**
  - \( P_i \) not in SB virtual, chosen page frame not in SB real

Summary

- **Reference patterns in DBS are hybrids**
  - sequential cyclic, random access
  - locality within and between transactions, “known” pages with high reference density
  - detection of scan-based processing
- **Without locality, optimization of page replacement is useless (~ RANDOM)**
- **Search in the buffer via hashing**
- **Memory allocation**
  - global \( \Rightarrow \) all buffer frames for all transactions (simplicity, stability, ...)
  - local \( \Rightarrow \) special handling of specific TAs/queries/ DB areas
- **Handling modified pages**: NoForce, asynchronous propagation
- **Page replacement algorithms**
  - prefetching and pipelining using alternating buffers
  - “too accurate” methods are difficult to configure (\( \Rightarrow \) unstable)
  - exploitation of several criteria: age, most recent reference, reference LRU, but simpler implementation; GCLOCK, LRD, LRU-K relatively complex
  - LRU-2 good trade-off, most recent but one reference determines “victim”
- **Extended replacement algorithms**
  - Use of access information of the query optimizer
  - Hot Set model
  - Multiple buffer use to separate loads of different types, optimized for specific data types
- **Double Paging should be avoided**