A One-Pass Aggregation Algorithm with the Optimal Buffer Size in Multidimensional OLAP

September 23, 2002

Young-Koo Lee, Kyu-Young Whang, Yang-Sae Moon, and Il-Yeol Song

Department of Computer Science and Advanced Information Technology Research Center(AITrc)

KAIST, Korea
Overview

- Introduction
- Motivation and Goals
- Computation Model Based on Disjoint-Inclusive Partition
- One-Pass Aggregation Algorithm and Its Optimality
- Experimental Results
- Conclusions
On-Line Analytical Processing: OLAP

- OLAP is a database application that allows users to easily analyze large volumes of data in order to extract information necessary for decision making.

- Example: Customer Data Analysis

  - Query Example:

    Find the total sales for each age

- Multidimensional OLAP: MOLAP
  - Uses multidimensional files as storage structures
Aggregation

- **Definition** (*Aggregation*):
  An operation that classifies records into groups and determines one value per group by applying the given aggregate function [Graefe 93]
  - *Grouping attributes*: the attributes used for grouping
  - *Aggregated attribute*: the attribute to which the aggregate function is applied

- **Examples**:
  - Find the total sales
  - Find the total sales for each year

- **OLAP queries make heavy use of aggregation for summarizing data**

- **Since computing aggregation is very expensive, good aggregation algorithms are crucial for achieving performance in OLAP systems**
Terminology

- **Organizing attributes**: a subset of attributes that determines the placement of records in the multidimensional file (i.e., attributes that correspond to dimensions)

- **Domain**: a set of values from which an attribute value can be drawn

- **Domain space**: the Cartesian product of all domains

- **Page region**: a region associated with a disk page

- **Grouping domain space**: the Cartesian product of the domains of all the grouping attributes

- **Grouping region**: any subset of the grouping domain space

- **Page grouping region**: the projection of the page region onto the grouping domain space
Related Work

- Aggregation using multidimensional arrays [Zhao et al. 97]
  - Stores data in a multidimensional array
  - Computes aggregation by accessing records in the unit of a page along the line perpendicular to the axis of the grouping attribute
  - Example: Aggregation of Y values for each X value in X-Y two dimensional space

(a) accessing in the unit of cells
(b) accessing in the unit of pages

- Is not efficient for skewed distributions
Our Approach

- To use a dynamic multidimensional file that handles skewed distributions efficiently
A Naïve Aggregation Method
Using a Dynamic Multidimensional File

Aggregation of Z values for each pair of X and Y values in a three dimensional space

- The aggregation is computed as the union of partial aggregations, each of which is computed for an aggregation window.

- Definition: *Aggregation windows* are grouping regions that form a partition of the grouping domain space and that are used to compute aggregation.

- Partial aggregation for an aggregation window is computed by retrieving records through a range query against the multidimensional file.
Problems

Aggregation of Z values for each pair of X and Y values in a three dimensional space

- The pages having large regions are accessed multiple times
- Example:
  - Page F (marked by blue color) is accessed twice since its page grouping region overlaps with two aggregation windows W3 and W4

: aggregation windows
Solution

- Use buffer

- Control the order of accessing pages to maximize the buffering effect
Buffer Replacement Policies

- **When the order of accessing pages is unknown**
  - The common strategy is to select the page that has the longest expected time until the next access
  - Examples: LRU [Coffman et al. 73], CLOCK [Effelsberg et al. 84], LRU-k [O’Neil et al. 93]

- **When the order of accessing pages is known in advance**
  - Belady’s B₀ [Coffman et al. 73]: selects as a victim the page that has the longest time until the next access
    - Proven to be the optimal buffer replacement policy
  - Toss-Immediate [Korth et al. 91]: upon each page access, immediately invalidates the page that will not be used further

- **Since the order of accessing pages is not known a priori in general, Belady’s B₀ and Toss-Immediate policies have been known to lack practicality**

⇒ Nevertheless, in this paper, we show that these policies can be effectively used for aggregation computation
Goals

- We propose an aggregation method that uses dynamic multidimensional files adapting to skewed distributions.
- We present a formal basis for aggregation computation, called the Disjoint-Inclusive Partition (DIP) computation model.
- We propose an aggregation method that maximizes the buffering effect by controlling the page access order.
- We formally prove that our algorithm achieves the optimal one-pass buffer size under the DIP computation model, which is the minimum buffer size required for one disk access per page.
Disjoint-Inclusive Partition

- When page regions and aggregation windows have certain topological relationships, we can improve the performance and buffering effect of computing aggregation by exploiting them

- Definition 1: Two regions $S_1$ and $S_2$ satisfy the disjoint-inclusive relationship if either $S_1$ and $S_2$ are disjoint or one includes the other

- Definition 2: A disjoint-inclusive partition (DIP) of the domain space $\mathbb{D}$ is a set $Q$ of regions satisfying the following conditions:
  (1) $Q$ is a partition of $\mathbb{D}$
  (2) When two regions in $Q$ are projected onto any subspace, the projected regions satisfy the disjoint-inclusive relationship

- Definition 3: We call a multidimensional file whose page regions form a DIP a DIP multidimensional file
Example: A DIP and a non-DIP

(a) A DIP.

(b) A non-DIP.

- Organizing attributes: X, Y, Z
- Set of grouping attributes G = \{X, Y\}

\(\Pi_G A\) and \(\Pi_G D\) (also \(\Pi_G A\) and \(\Pi_G E\)) do not satisfy the disjoint-inclusive relationship.
DIP Computation Model

Definition: The *DIP computation model* for computing aggregations using a multidimensional file is the one that satisfies the following four conditions:

1. It uses a DIP multidimensional file
2. The aggregation for the grouping domain space is computed as the union of partial aggregations for aggregation windows
3. Disjoint-inclusive relationship is satisfied among aggregation windows and page grouping regions
4. Each partial aggregation is computed by retrieving records through a range query against the multidimensional file
Controlling the Order of Accessing Pages

- **Definition (L-page):**
  
  A page $P$ is an $L$-page (large page) of an aggregation window $W_i$ if the page grouping region of $P$ properly includes $W_i$.

- **Objective**
  
  - To make an L-page be accessed from disk only once by accessing the pages in a specific order.

- **For this specific order, we propose an optimal space filling curve, called Induced Space Filling Curve, based on the formal properties of DIP.**
Induced Space Filling Curve (ISFC)

- **Definition (Induced Space Filling Curve (ISFC))**: A space filling curve induced from a given set of regions so that it can traverse all smaller regions included in a region $S_i$, and then, traverse those that are not included in $S_i$

- **Lemma 2**: For a given set $S$ of regions, where elements of $S$ satisfy the disjoint-inclusive relationship, there exists at least one ISFC

- **Definition ($ISFC_\mathcal{R} \cup \mathcal{W}$)**: ISFC based on the given set $\mathcal{R} \cup \mathcal{W}$
  - $\mathcal{R}$: a set of page grouping regions in a DIP multidimensional file
  - $\mathcal{W}$: a set of aggregation windows

- **Lemma 3**: When traversing the aggregation windows in $ISFC_\mathcal{R} \cup \mathcal{W}$ order, L-pages are accessed in contiguous aggregation windows
Example: An ISFC_{R \cup W} and a non-ISFC_{R \cup W}

(a) Page grouping regions (R_i’s).

(b) Aggregation windows (W_i’s).

(c) An ISFC_{R \cup W}.

(d) A non-ISFC_{R \cup W}.

Traversing order for W_i’s: W_1 \rightarrow W_2 \rightarrow W_4 \rightarrow W_3
The One_Pass_Aggregation Algorithm

- Uses DIP computation model

- Traverses aggregation windows in an ISFC_\( R \cup W \) order

- Uses Belady’s \( B_0 \) or Toss-Immediate as the buffer replacement policy
Toss-Immediate Policy

- When traversing the aggregation windows in the order of \( ISFC_{R \cup W} \), we can identify in Lemma 4 the page that will no longer be accessed.
  \[ \Rightarrow \text{Toss-Immediate} \]

- Lemma 4: A page \( P_{curr} \) accessed during partial aggregation for \( W_{curr} \) will no longer be accessed if it satisfies the following condition:
  - The \( ISFC_{R \cup W} \) value of \( W_{curr} \) is greater than or equal to that of the page grouping region of \( P_{curr} \)

- Similarly, we can use Belady’s \( B_0 \)
  - Details are referred to a future paper
Algorithm One_Pass_Aggregation

Input:
(1) DIP multidimensional file *md-file* that contains OLAP data
(2) Set \( G \) of grouping attributes
(3) Aggregated attribute \( A \)

Output:
Result of aggregation

1 Partition the grouping domain space into aggregation windows so that aggregation windows and page grouping regions satisfy the disjoint-inclusive relationship.

2 Initialize the buffer:
   2.1 Compute the one-pass buffer size, \( BUFSIZE = \max_{W_i} \{ \text{(the number of L-pages of } W_i') + \alpha_i \} \)
   2.2 Allocate the buffer of size \( BUFSIZE \)

3 Traversing aggregation windows in \( ISFC \cup W \) order, for each aggregation window \( W_{curr} \), DO
   3.1 Construct a range query. Here the query region consists of the intervals corresponding to the aggregation window for the grouping attributes and the entire domain of each attribute for the other organizing attributes.
   3.2 Process the range query against *md-file*
      3.2.1 While evaluating the range query, when processing the current page \( P_{curr} \) is completed, if \( ISFC_{\cup W} (\Pi_G P_{curr}) \leq ISFC_{\cup W} (\Pi_G W_{curr}) \), then remove \( P_{curr} \) from the buffer.
      3.2.2 For each record retrieved, using the values of the attributes in \( G \), find the corresponding entry from the window result table, aggregate the value of the attribute \( A \), and store the result into the entry.
Optimal One-Pass Buffer Size under DIP Computation Model

• **Definition:** The *one-pass buffer size* is the minimum buffer size required for guaranteeing one disk access per page.

• **Theorem 1 (and 2):** The optimal one-pass buffer size under the DIP computation model is $\max_{W_i} \{(\text{the number of L-pages of } W_i) + \alpha_i\}$
  - $\mathcal{W} = \{W_1, W_2, \ldots, W_k\}$ is a set of aggregation windows.
  - $\alpha_i$ is 1 if at least one aggregation window page of $W_i$ is a non-L-page and if it is accessed after all the L-pages of $W_i$ have been read into the buffer; $\alpha_i$ is 0 otherwise.
Proof (intuitive):

- The buffer should contain as many pages as the number of L-pages.
- By traversing the aggregation windows in the order of $\text{ISFC}_{R \cup W}$, L-pages are accessed in contiguous aggregation windows (Lemma 3).
- By using the Toss-Immediate policy, the L-page contiguously accessed once is immediately invalidated and will not be accessed further.
- In addition, one page is needed to fetch a non-L-page into the buffer if it exists.
- Detailed proof is in the paper.
Optimality of One_Pass_Aggregation Algorithm

Theorem 2: The one-pass buffer size of the One_Pass_Aggregation algorithm is equal to the optimal one-pass buffer size of the DIP computation model

Proof: See the paper
Experiments

- Objectives

  - We demonstrate that the one-pass buffer size theoretically derived is indeed correct in real environments
  
  - We compare the performance of the one-pass algorithm with those of other ones
  
  - We show that our algorithm requires a relatively small amount of main memory for processing the aggregation in one pass
**Algorithms Compared:**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Conformance to DIP computation model</th>
<th>Traversal order for aggregation windows</th>
<th>Buffer replacement policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve_Aggregation</td>
<td>NO</td>
<td>Row-major</td>
<td>CLOCK</td>
</tr>
<tr>
<td>DIP_Aggregation</td>
<td>YES</td>
<td>Hilbert</td>
<td>CLOCK</td>
</tr>
<tr>
<td>ISFC_Aggregation</td>
<td>YES</td>
<td>ISFC&lt;sub&gt;RUW&lt;/sub&gt;</td>
<td>CLOCK</td>
</tr>
<tr>
<td>One_Pass_Aggregation</td>
<td>YES</td>
<td>ISFC&lt;sub&gt;RUW&lt;/sub&gt;</td>
<td>Toss-Immediate</td>
</tr>
</tbody>
</table>
Experimental environment

- **Multidimensional file used:**
  
  MLGF [Whang and Krishnamurthy 85]

- **Data sets:** Records consisting of six attributes (five attributes: dimensions, one attribute: measure). Three among five dimensions are used as grouping attributes
  
  1) SYNTHETIC-DATA: a synthetic data set with a distribution that superposes 100 overlapping multivariate normal distributions simulating multiple clusters
     - SMALL-DATA: 500,000 records (29.9MB)
     - MEDIUM-DATA: 5,000,000 records (279.3MB)
     - LARGE-DATA: 50,000,000 records (2,648.3MB)
  
  2) REAL-DATA: the Forest Cover Type data with about 600,000 records (http://kdd.ics.uci.edu/)
• Measures

  • Normalized I/O accesses = \frac{\text{Number of page accesses}}{\text{Total number of pages in the file}}

  • Main memory requirement
The one-pass buffer size computed experimentally turns out to be equal to that predicted in Theorem 2

- One_Pass_Aggregation > ISFC_Aggregation > DIP_Aggregation > Naïve_Aggregation
Normalized I/O Accesses (REAL-DATA)

- **Performance gaps are more marked in REAL-DATA**
  - There is more random variation in data distribution for real data than for synthetic data
Memory Requirement of the One_Pass_Aggregation Algorithm

- Total main memory requirement = one-pass buffer size + window result table size

- The window result table is memory used to contain the result of partial aggregation for an aggregation window

![Graphs showing memory requirement for SMALL-DATA and MEDIUM-DATA](attachment:graphs.png)
The memory requirement for processing the aggregation in one pass is very small being 0.05% ~ 0.60% of the size of the database.
Conclusions

- We have presented an aggregation algorithm that uses dynamic multidimensional files adapting to skewed distributions.
- We have presented the new notion of the disjoint-inclusive partition (DIP) and proposed a formal basis for aggregation computation, called the DIP computation model.
- We have formally derived the optimal one-pass buffer size under the DIP computation model (Theorems 1 and 2).
- We have proposed the One_Pass_Aggregation algorithm that utilizes the page access order computed in advance.
- We have proven that the One_Pass_Aggregation algorithm has the optimal one-pass buffer size.
- We believe we have provided an excellent formal basis for investigating further issues in computing aggregations in MOLAP.