Vertical Partitioning for Query Processing over Raw Data

Weijie Zhao, Yu Cheng, Florin Rusu
Electrical Engineering and Computer Science, University of California, Merced
wzhao23, ycheng4, frusu@ucmerced.edu

Query Processing over Raw Data

The generic procedure to extract tuples with the given schema from the raw file

- **Query Engine**
  - EXTRACT
    - Tokenize
    - Parse
  - READ
  - WRITE
  - Disk

Mixed Integer Programming

\[
\begin{align*}
\text{minimize} & \quad T_{\text{load}} + \sum_{i=1}^{m} w_i \cdot T_i, \quad T_i \text{ subject to constraints:} \\
C_1: & \quad \sum_{j=1}^{n} \text{save}_j \cdot SPF_j \cdot |R| \leq B \\
C_2: & \quad \text{read}_j \leq \text{save}_j; \quad i = 1, m, j = 1, n \\
C_3: & \quad \text{save}_j \leq \text{raw}_j; \quad i = 1, m, j = 1, n \\
C_4: & \quad \text{raw}_j \leq \text{raw}_j; \quad i = 1, m, j = 1, n \\
C_5: & \quad T_j \cdot \text{read}_j + \text{save}_j = 1; \quad i = 1, m, j = 1, n, A_j \in Q, \\
\end{align*}
\]

Execution time of a query \(T_i\) is a slight modification:

\[
T_i = \text{raw}_i \cdot \frac{S_{\text{raw}}}{\text{band}} + |R_i| \cdot \sum_{j=1}^{n} \left( t_{ij} \cdot T_j + p_{ij} \cdot T_p + \text{save}_j \cdot \frac{SPF_j}{\text{band}} \right)
\]

Query Coverage

- **Input:** Workload \(W = (Q_1, \ldots, Q_m)\); storage budget \(B\)
- **Output:** Set of attributes \((A_1, \ldots, A_k)\) to be loaded in processing representation
  1. \(attsL = \emptyset, \text{coveredQ} = 0\)
  2. while \(\sum_{j \in attsL} \text{SPF}_j < B\) do
  3. \(idx = \text{argmax}_{j \in \text{coveredQ}} \left( \frac{\text{cost}(attsL) - \text{cost}(attsL \cup Q_{idx})}{\sum_{j \in \text{coveredQ}} \text{SPF}_j} \right)\)
  4. if \(\text{cost}(attsL) - \text{cost}(attsL \cup Q_{idx}) \leq 0\) then break
  5. \(\text{coveredQ} = \text{coveredQ} \cup \text{idx}\)
  6. \(attsL = attsL \cup Q_{idx}\)
  7. end while
  8. return \(attsL\)

Attribute Usage Frequency

- **Input:** Workload \(W = (Q_1, \ldots, Q_m)\) of \(R\); storage budget \(B\); set of loaded attributes \(\text{saved} = (A_1, \ldots, A_n)\)
- **Output:** Set of attributes \((A_{r1}, \ldots, A_{rn})\) to be loaded in processing representation
  1. \(attsL = \text{saved}\)
  2. while \(\sum_{j \in attsL} \text{SPF}_j < B\) do
  3. \(idx = \text{argmax}_{j \in \text{attsL}} \left( \text{cost}(attsL) - \text{cost}(attsL \cup A_j) \right)\)
  4. \(attsL = attsL \cup \text{idx}\)
  5. end while
  6. return \(attsL\)

Heuristic Algorithm

- **Input:** Workload \(W = (Q_1, \ldots, Q_m)\); storage budget \(B\)
- **Output:** Set of attributes \((A_1, \ldots, A_k)\) to be loaded in processing representation
  1. \(\text{obj}_{\text{min}} = \infty\)
  2. for \(i = 0; i < B\) do
  3. \(attsL_i = \text{Query coverage}(W, i)\)
  4. \(attsL_i = \text{Attribute usage frequency}(W, \Delta_i \cdot attsL_{<i})\)
  5. \(attsL = attsL_i \cup attsL_{<i}\)
  6. \(\text{obj} = \text{cost}(attsL)\)
  7. if \(\text{obj} < \text{obj}_{\text{min}}\) then
  8. \(\text{obj}_{\text{min}} = \text{obj}\)
  9. \(attsL_{\text{min}} = attsL\)
  10. end if
  11. end for
  12. return \(attsL_{\text{min}}\)

Experiments

- **Model validation:** serial CSV, serial FITS, pipeline JSON.
- **Impact of Each Stages of Heuristic on CSV**
- **Sequential Processing on FITS**
- **Pipeline Processing on JSON**

Weijie Zhao, Yu Cheng, Florin Rusu (EECS, University of California, Merced)