Approximate Join Processing Over Data Streams

Abhinandan Das, Johannes Gehrke, Miek Riedewald @ Cornell
In SIGMOD 2003

Presented by Feifei @ Cs.Bu
Data Stream and Join Operator

- **Data Stream Processing Systems vs. Traditional DBMS**
- **Sliding Window Join**
  - Join the set of tuples in a bounded-size window of the most recent items in two streams $r(i)$ and $s(i)$, $t-w < i \leq t$
- **Resource Limitations**
  - Fast CPU vs. Slow CPU (high arriving rate)
  - Memory Restriction (Q: what’s the minimum memory requirement for exact join?)
Semantic Load Shedding

- Resource Limitations – how to solve it?
  - Drop some tuples to get approximation!
- Random Load Shedding vs. Semantic Load Shedding
  - Random Load Shedding – Easy to see
  - Approximate the output of an operator by maximizing a user-defined similarity measure between the exact and the approximation
Join Processing Models

- Fast CPU vs. Slow CPU
Error Measures

- MAX-Subset Measure
  - The Symmetric Difference: $|(X - Y) \cup (Y - X)|$
  - For equi-joins, dropping tuples before they expire naturally leads to the generated output being a subset of the exact join result.
  - What the symmetric difference in this case implies?

- Archive-Metric (ArM) for Join with Archive Support
  - Observation: Streams are not always arriving at a consistently high rate!

\[
\delta_R(i, j) = \begin{cases} 
1 & \text{If } r(i) \text{ survived for at least } j \text{ time in memory}, \\
0 & \text{otherwise}.
\end{cases}
\]

\[
S^<(i) = \{ j : j \in [i - w + 1, i - 1] \land s(j) = r(i) \}
\]

\[
j_{r(i)} = \max \{ j : j \in [i, i + w - 1] \land s(j) = r(i) \}
\]

\[
\sum (\delta_R(i, j_{r(i)} - i) \prod_{j \in S^<(i)} \delta_S(j, j - i) + \delta_S(i, j_{s(i)} - i) \prod_{j \in R^<(i)} \delta_R(j, j - i))
\]
MAX-Subset Measure – static case

- Aim: Find a set of $k$ tuples to be dropped from the input relation s.t. the size of the $k$-truncated join result is as large as possible.
- Model as a graph problem, bipartite graph $G(Va, Vb, E)$
  - $Va Vb$ represent the two relations, each has one node for every tuple, an edge exists between two nodes if the corresponding tuples satisfy the join condition.
  - $G$ will consist of a union of mutually disjoin fully connected bipartite components (called Kurotowski components), represented as $k(m,n)$.
MAX-Subset Measure – static case

- **Input**: A bipartite graph consisting of $c$ mutually disjoint Kurotowski subgraphs by the $c$ integer pairs $K(m_1,n_1), K(m_2,n_2)\ldots$
- **Output**: A set of $k$ nodes to be retained s.t. the subgraph induced by them has the highest number of edges among all subgraphs with $k$ nodes.

![Graph with nodes and edges](image)
MAX-Subset Measure – static case

- An Optimal Dynamic Programming Solution
  - Given \( k(m,n) \), \( m' \times n' \) as large as possible, so \( m'+n'=p \) and \( |m'-n'| \) as small as possible
  - \( P \) nodes can be retained by chosen one by one from each partition. \( C_{m,n}(p) \) “max num. of edges can be retained when \( p \) nodes are retained from \( k(m,n) \)

\[
C_{m,n}(p) = \begin{cases} 
(p/2)^2 & \text{if } p \leq 2n, p \text{ even} \\
(p^2 - 1)/4 & \text{if } p \leq 2n, p \text{ odd} \\
n(p-n) & \text{else.}
\end{cases}
\]

\[
T(1, j) = \begin{cases} 
C_{m_1,n_1}(j) & \text{if } 0 \leq j \leq m_1 + n_1 \\
-\infty & \text{if } j > m_1 + n_1
\end{cases}
\]

\[
T(i, j) = \max \begin{cases} 
T(i-1, j), \\
T(i-1, j-1) + C_{m_t,n_t}(1), \\
T(i-1, j-2) + C_{m_t,n_t}(2), \\
\vdots \\
T(i-1, j-m_t-n_t) + C_{m_t,n_t}(m_t+n_t)
\end{cases}
\]

How about 3-relation static join load shedding problem?
MAX-Subset Measure - Fast CPU and offline

- OPT-Offline Algorithm based on Flow Graph
- The Flow Graph: \( R = 1,1,1,3,2 \) \( S = 2,3,1,1,3 \), M=2, w=3

(r(1), s(2)), (r(1), s(3)) missed!
MAX-Subset Measure - Fast CPU and offline

- Solving the linear minimum-cost flow problem for this graph produces the optimal strategy for deciding which tuples to drop from memory at each time instant!

- The highest absolute arc cost in network is 1, known algorithms find the optimal integer solution in time $O(n^2 m \log n)$, $m$ is num of arcs, $n$ is num of nodes

- $N$: num of tuples in each stream, then total nodes will be $2wN+N+2$, total edges will be: $(M+1+3 \times \text{numberNodes}-2)$
MAX-Subset Measure - Fast CPU and online

- PROB Heuristic
  - PROB estimates for each value in the domain of the join attribute the probability of a tuple with this value arriving on stream R and S.
  - The priority for r(i) is $Ps(r(i))$, ties are broken by giving higher priority to the tuple that arrived later.
  - Favors high partner arrival probabilities over age.
  - How to compute $Pr()$ and $Ps()$ without knowing the future?

- LIFE Heuristic
  - Priority is $t*Ps(r(i))$, $t$ is the remaining lifetime of $r(i)$.

Which one is better?
Experiments

Figure 3: Window size 400

Figure 4: Window size 800
Experiments

Figure 5: Uniform input
Experiments

Figure 6: Uncorrelated Zipf
Experiments

Figure 9: Domain size 10

Figure 11: Domain size 200
Conclusion

- Carefully designed Semantic Load Shedding Algorithm is much better than random load shedding

- Problems:
  - How about Archive-metric?
  - Slow CPU?
  - Even more streams?
Thanks!