An Intersection Cache Based on Frequent Itemset Mining in Large Scale Search Engines

Wanwan Zhou, Ruixuan Li, Xinhua Dong, Zhiyong Xu, Weijun Xiao

Huazhong University of Science and Technology
Wuhan, China

Modern Information Retrieval

**Characteristics:** mass data and large query requests

<table>
<thead>
<tr>
<th>Type</th>
<th>Capacity(GB)</th>
<th>Read (MB/S)</th>
<th>Write (MB/S)</th>
<th>Unit Price(RMB/GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD</td>
<td>500</td>
<td>100</td>
<td>80</td>
<td>0.56</td>
</tr>
<tr>
<td>RAM</td>
<td>5</td>
<td>6700</td>
<td>5700</td>
<td>92.5</td>
</tr>
</tbody>
</table>

**Bottleneck:** I/O

Distributed system, Index compression, **Cache**, etc

Wanwan Zhou, Ruixuan Li, Xinhua Dong, et al.,
Huazhong University of Science and Technology
Related Work

Multi-level Caching Architecture
- Three-level cache architecture based on intersection cache on HDD
- Five-level static cache architecture

Intersection Cache Strategy
- Static projection cache - Greedy Offline (WWW)
- Dynamic projection cache - Landlord Online (WWW)
- Integrated list (SPIRE)
- FTRC (Full-Term-Ranking-Cache) & TTIC (Two-Term-Intersection-Cache) (TOIS)

Frequent Itemset Mining Algorithms
- Apriori & FP-Growth
- FP-Tree & Trie-Tree
### Problems and Motivation

- Explosive intersection data growth
- Limited to two-terms pairs
- Low efficiency of static intersection data selection strategy
- Serious delay in dynamic intersection cache strategy

Explore suitable intersection cache strategies to achieve a good balance among the retrieval performance, cache data strategy efficiency and application flexibility.

### Contributions

1. Analyzing the search engine query log and identifying intersection characteristics.
2. Proposing a three-level memory cache architecture TLMCA.
3. Presenting an intersection cache data selection strategy based on frequent itemset mining.
4. Proposing an intersection cache data replacement strategy based on incremental frequent itemset mining.
Intersection characteristic analysis

- The query, query term and intersection all follow the Power-Law distribution with $\alpha$ equals 0.73, 1.1 and 0.76 respectively.
Intersection characteristic analysis

- The length of query is mainly less than or equal to 3.
- With the increase of k, the maximum support of k-itemsets are in non-ascending order.
- The hit count of intersection is relatively low and the number of hit count is one in most cases.
- Users’ queries meet the closure properties of frequent itemsets.

Three-Level Memory Cache Architecture

[Diagram of the Three-Level Memory Cache Architecture]

- Query is sent to the Result Cache.
- If a hit is found, the result is returned.
- If not, the query is sent to the Index Servers for further processing.
- The Index Servers retrieve the relevant documents and generate snippets.
- The Result Cache combines the snippets to form the final result.

Wanwan Zhou, Ruixuan Li, Xinhua Dong, et al.
Huazhong University of Science and Technology
Outline

- Characteristic analysis
- Data selection strategy
- Data replacement strategy
- Experiments
- Conclusion

ICSS_FIMI

**Input:** maxLength, N, query log
**Output:** Top-N intersections L

1. Build a complete FP-Tree
   - Calculate the support of every term
   - Reorder every query
   - Build a complete FP-Tree

2. Set $Sup_{min}$
   - Calculate the support of every term pair
   - $Sup_{min}$ = the support of the Nth term pair
ICSS_FIMI

3. Form a new FP-Tree
   • Prune, and delete the node from the FP-Tree whose support less than \( Sup_{min} \).

4. Figure out frequent itemsets
   • Build a conditional pattern base (CPB) of every node on the FP-Tree, and calculate combinations and their support.

5. Output result L
   • Output Top-N intersections, and return.

---

ICSS_FIMI

Time complexity analysis:

\[
P = O \left[ T + N^2 \times \log N + 2L + logL + M \times (T + 2) + N \times \left( T + \sum_{k=2}^{\text{max length}} q_k \right) + X \times \log X \right]
\]

\[
= O \left( N^2 \times \log N + M + T \right)
\]

- \( T \): the number of queries;
- \( M \): the number of different terms;
- \( L \): maximum length of a query;
- \( X \): the number of intersections.
Outline

- Characteristic analysis
- Data selection strategy
- Data replacement strategy
- Experiments
- Conclusion

ICRS_IFIMI

**Input:** maxLength, N, query log of slide windows

**Output:** Top-N intersections L

1. Slide window initialization
2. Update slide window and Trie-Tree
3. Set $Sup_{min}$
4. Figure out frequent itemsets
5. Output result L
Outline

- Characteristic analysis
- Data selection strategy
- Data replacement strategy
- Experiments
- Conclusion

Experiments and evaluation

<table>
<thead>
<tr>
<th>Test-platform Environment</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IR Tool</td>
<td>Lucene 3.0.0</td>
</tr>
</tbody>
</table>
| Data set                 | Enwiki-20090805-pages-articles.xml  
5 million docs, index file 5.2GB |
| Query log                | AOL-user-ct-collection, 3,519,003 queries  
1,197,567 different queries, 580,116 query terms, and 6,535,327 intersections, The average length of a query is 2.23, and the longest query contains 132 terms. |
| CPU/RAM                  | Intel Core2 Duo P8600(2.40GHz;1066MHz;3072KB)/4GB |
| OS                       | Window 7/ Ubuntu 10.04 |
| HDD                      | HITACHI HTS545025B9A300 250GB |

Retrieval performance
Cache hit ratio
Intersection cache strategy efficiency
Retrieval performance

Cache hit ratio

IC has little impact on the hit ratio of RC or PLC.
Intersection cache strategy efficiency

- Strategies based on FIMI and Trie-Tree are more efficient respectively.

Outline

- Characteristic analysis
- Data selection strategy
- Data replacement strategy
- Experiments
- Conclusion
Conclusion and future work

- The experimental results demonstrate our design
  - IC can improve the retrieval performance in isolation, but its effect is less than RC or PLC.
  - Compared to 2L_RC+PLC, 3L_RC+PLC+IC improves the retrieval performance by 27%.
  - A few IC has no impact on the hit ratio of RC or PLC.
  - IC strategies based on FIMI are much more efficient.

- The future work
  - Consider IC update strategy in dynamic scenario.
  - Cache strategy or index structure optimization based on In-Memory Computing technology.

Thanks for your attention

- Contact information:
  - Ruixuan Li
  - Huazhong University of Science and Technology
  - rxli@hust.edu.cn
  - http://idc.hust.edu.cn/~rxli/