DATA CACHING IN WSN

Mario A. Nascimento
Univ. of Alberta, Canada
http://www.cs.ualberta.ca/~mn

With R. Alencar and A. Brayner. Work partially supported by NSERC and CBIE (Canada) and CAPES (Brazil)
Outline

- Motivation
- Cache-Aware Query Processing
- Cache-Aware Query Optimization
  - Query Partitioning
  - Cached Data Selection
- Cache Maintenance
- Experimental Results
(One) Application Scenario

User → Base Station → Satellite → WSN

User → Base Station

User → Base Station
Using Previous (Cached) Queries

\{ P_i \} : Set of previous queries

\( Q \) : Current query

\( Q' \) : \( Q \) “minus” \( \{ P_i \} \)
Query Partitioning Overhead

Query Processing: query is forwarded, locally flooded, results are collected and shipped back

Query processing cost is estimated through an analytical cost-model
Overall Architecture

User

Current query

Q → D(Q)

Answer

Q, P', D(P') → Θ

WSN

Subset of relevant queries and sub-queries (min: query cost)

Cache Manager

Query Processor

Query Optimizer

Base Station

Cache Index

Relevant Cached Queries

Non-stale subset of P and its dataset
“Query Plan” Problem (QSP)

Less larger sub-queries vs. more smaller sub-queries

For obtaining $Q'$ we used the General Polygon Clipper library. For partitioning $Q'$ into the set of sub-queries $\Theta$ we used a $O(v \log v)$ algorithm which finds a sub-optimal solution (minimizing the number of sub-queries).
**B+B (Heuristic) Solution to QSP**

For each node, each node is "clipped" using a subset of $P''$, a set of sub-queries is generated and its cost is obtained. The search stops at a local minimum.
Other Heuristic Solutions to QSP

In addition to the B+B we also used two more aggressive greedy heuristics: GrF (GrE) starts with all (no) cached queries removing (inserting) the smallest (largest) cached query as long as there is some gain.

GrF “path”
Cache Maintenance

Cache Reader

Query Processor

Cache Updater

Cache Manager (internals)

Cache Index

$Q$

$P', D(P')$

$P', P'', \Theta$

$P \setminus P'$

$P$

$Q, P \setminus P', P' \setminus P'', \Theta$
Cache Maintenance

Data that can be used to refresh $P_1$’s data

$P_1$ (dropped)

$P_2$ (used)

$Q$

$Q'$

$P_{1,1}$

$P_{1,2}$

$P_3$

$P_2$
B+B is the Branch-and-Bound heuristic. GrF (GrE) is an aggressive greedy heuristic, starting with all (no) cache and removing (inserting) the smallest (largest) cached queries available as long as there is some gain.
By design GrE cannot be any worse than no using any cache.
Gains wrt Using ALL Cache

By design GrF cannot be any worse than using all of the cache.
• Detailed results or skip to main conclusions?
Detailed results

We investigate the performance of the proposed approach wrt efficiency (for finding the query plan) and effectiveness (cost of solution) when varying:

- Number of sensors
- Size of cache (number of cached queries)
- Query size (wrt total area)
- Validity time (of cached results)
Varying # of Sensors

Energy cost loss wrt OPT [%] vs Number of sensors (x 1,000)

Number of states explored vs Number of sensors (x 1,000)
Varying Cache Size

![Graph 1: Energy cost loss wrt OPT (%) vs Cache size (# Queries)]

- B+B
- FC
- GrF
- GrE

![Graph 2: Number of states explored vs Cache size (# Queries)]

- GrE
- GrF
- B+B
- OPT
Varying Query Size

Energy cost loss wrt OPT [%]

Query size [% of total area]

Number of explored states

Query size [% total area]
Varying Query Validity Time

Energy cost loss wrt OPT [%]

Validity time [number of timestamps]

Number of states explored

Validity time [# timestamps]
Conclusions

- The cached query selection, query clipping and sub-queries generation amounts to a fairly complex and combinatorial problem.

- Although a query cost model is needed, our proposal is orthogonal to it.

- If nothing can be done your best shot is to use all of the cache, but …
Conclusions

- The Branch-and-Bound heuristic:
  - Finds a “query plan” orders of magnitude faster than the exhaustive search
  - Is typically less than 2% more expensive than the optimal query cost
  - Is robust with respect to a number of different parameters

- Next stop:
  - Aggregation queries …
Thanks