An Evaluation of Multi-resolution Storage for Sensor Networks

Deepak Ganesan, Ben Greenstein, Denis Perelyubskiy, Deborah Estrin (UCLA), John Heidemann (USC/ISI)

Presenter: Vijay Sundaram
Constructing the hierarchy

Initially, nodes fill up their own storage with raw sampled data.
Constructing the hierarchy

- Tesselate the network space into grids, and hash in each to determine location of clusterhead (ref: DCS).
- Send wavelet-compressed local time-series to clusterhead.
Processing at each level

... Store incoming summaries locally for future search.

Get compressed summaries from children.

Decode

Wavelet encoder/decoder

Re-encode at lower resolution and forward to parent.
Constructing the hierarchy

Recursively send data to higher levels of the hierarchy.
Distributing storage load

Hash to different locations over time to distribute load among nodes in the network.
What happens when storage fills up?

Eventually, all available storage gets filled, and we have to decide when and how to drop summaries.

Allocate storage to each resolution and use each allocated storage block as a circular buffer.
Tradeoff between Age and Storage requirements for summary

**Graceful Query Degradation:** Provide more accurate responses to queries on recent data and less accurate responses to queries on older data.

How do we allocate storage at each node to summaries at different resolutions to provide gracefully degrading storage and search capability?
Match system performance to user requirements

User provides a function, $Q_{user}$, that represents desired query quality degradation over time.

System provides a step function, $Q_{system}$, with steps at times when summaries are aged.

Objective: Minimize worst case difference between user-desired query quality (blue curve) and query quality that the system can provide (red step function).
What do we know?

Given

- \( N \) sensor nodes.
- Each node has storage capacity, \( S \).
- Users ask a set of typical queries, \( T \).
- Data is generated at resolution \( i \) at rate \( R_i \).
- \( D(q,k) \) – Query Error when drilldown for query \( q \) terminates at level \( k \).
- \( Q_{user} \) - User-desired quality degradation.
Determining Query Quality from multiple queries

We need to translate the performance of different Drill-down queries to a single "query quality" metric.
Definition: Query Quality

Given:
- \( T = \) set of typical queries.
- \( D(q,k) = \) Query error when drill-down for query \( q \) in set \( T \) terminates at resolution \( k \).

The query quality for queries that refer to data at time \( t \) in the past, \( Q_{system}(t) \), if \( k \) is the finest available resolution is:

\[
Q_{system}(t) = \frac{1}{|T|} \sum_{q \in T} D(q, k)
\]
How many levels of resolution, \( k \) are available at time \( t \) ?

Given:
- \( R_i \) = Total transmitted data rate from level \( i \) clusterheads to level \( i+1 \) clusterheads.

Define \( s_i \) = Storage allocated to each node for summaries at resolution \( i \).

\[
\text{Age}_i = \frac{Ns_i}{R_i}
\]
Storage Allocation: Constraint-Optimization problem

**Objective**: Find \( \{s_i\} \), \( i=1..\log_4 N \)

that:

\[
\min_{t = -\infty} \max_{0} Q_{\text{user}}(t) - Q_{\text{system}}(t)
\]

**Given constraints**:

- **Storage constraint**: Each node cannot store any greater than its storage limit.
- **Drill-down constraint**: It is not useful to store finer resolution data if coarser resolutions of the same data is not present.

\[
\sum_{i=1}^{\log_4 N} s_i \leq S
\]

\[
\text{Age}_{i+1} \geq \text{Age}_i
\]
How do we determine communication rates to, say, bound query error?
Assume: Rates are fixed a-priori by communication constraints.

How do we determine the drill-down query error when prior information about deployment and data is limited?
Prior information about sampled data

- **Omniscient Strategy**: Baseline. Use all data to decide optimal allocation.
- **Training Strategy**: (can be used when small training dataset from initial deployment).
- **Greedy Strategy**: (when no data is available, use a simple weighted allocation to summaries).

Solve Constraint Optimization

<table>
<thead>
<tr>
<th>Coarse</th>
<th>Finer</th>
<th>Finest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 :</td>
<td>2 :</td>
<td>4</td>
</tr>
</tbody>
</table>

No a priori information

full a priori information
Distributed trace-driven implementation

- Linux implementation for ipaq-class nodes
  - uses Emstar (J. Elson et al), a Linux-based emulator/simulator for sensor networks.
  - 3D Wavelet codec based on freeware by Geoff Davis available at: http://www.geoffdavis.net.
  - Query processing in Matlab.

- Geo-spatial precipitation dataset
  - 15x12 grid (50km edge) of precipitation data from 1949-1994, from Pacific Northwest†. (Caveat: Not real sensor data).

- System parameters
  - Training set: 6% of total dataset.

† M. Widmann and C. Bretherton. 50 km resolution daily precipitation for the Pacific Northwest, 1949-94.
How efficient is search?

Search is very efficient (<5% of network queried) and accurate for different queries studied.
Comparing Aging strategies

Training performs within 1% to optimal. Careful selection of parameters for the greedy algorithm can provide surprisingly good results (within 2-5% of optimal).
Future Research Directions

- Structured node placement (regular/grid) may not be possible due to terrain conditions.
- Extend current techniques to use wavelet schemes that handle irregularity †.
- Extend training schemes to be online.
- Implementation on motes.

Node placement at James Reserve

† Debauchies, Guskov, Schroder, Sweldons: Wavelets on irregular point sets
Progressive aging of summaries can be used to support long-term spatio-temporal queries in resource-constrained sensor network deployments.

We describe two algorithms: a training-based algorithm that relies on the availability of training datasets, and a greedy algorithm can be used in the absence of such data.

Our results show that

- training performs close to optimal for the dataset that we study.
- the greedy algorithm performs well for a well-chosen summary weighting parameter.