Cloud-based Query Evaluation for Energy-Efficient Mobile Sensing

Tianli Mo*, Sougata Sen†, Lipyeow Lim*, Archan Misra†, Rajesh Krishna Balan † and Youngki Lee†
*University of Hawaiʻi at Mānoa and †Singapore Management University

***Abstract*—In this paper, we reduce the energy overheads of continuous mobile sensing for context-aware applications that are interested in *collective context or events*. We propose a cloud-based query management and optimization framework, called *CloQue*, which can support concurrent queries, executing over thousands of individual smartphones. *CloQue* exploits correlation across context of different users to reduce energy overheads via two key innovations: i) dynamically reordering the order of predicate processing to preferentially selects predicates with not just lower sensing cost and higher selectivity, but that maximally reduce the uncertainty about other context predicates; and ii) intelligently propagating the query evaluation results to dynamically update the confidence values of other correlated context predicates. An evaluation, using real cellphone traces from a real world dataset shows significant energy savings (between 30 to 50% compared with traditional short-circuit systems) with little loss in accuracy (5% at most).**

I. Introduction

This work proposes a system for efficiently executing *multi-person, continuous* queries, expressed over context derived from smartphone-embedded sensors of a *large group* of individuals. In many context-aware computing scenarios, users are interested in context or events that are not just derived from a single individual, but are instead based on the collective context of a group. For example, a university student may wish to be notified when the rest of her project mates have reached a meeting room. Evaluation of such continuous, multi-person queries will often aggravate the energy overhead.

It is possible to reduce energy overheads in such evaluations by designing a technique which considers: i) *Correlation Across Users:* Users often perform activities in coordinated or correlated fashion and ii) *Sensor Diversity:* Different context attributes constituting a collective query require data from different sensors, thus affecting sensing costs. Evaluating "cheaper" sensors first can reduce the overall energy cost.

Both of the above strategies for query optimization have been investigated previously (e.g., context correlation in [8] and short-circuiting of queries in [4], [7]), but almost exclusively for retrieving context of an individual user in isolation. Our intention is to utilize the principles of query short-circuiting and context correlation to make evaluation of context more energy-efficient, but for collective context queries, *at scale*–e.g., over hundreds or thousands of individuals in environments such as office buildings or college campuses. Such a setting gives rise to several unique challenges such as: (i) *Varying levels of Cross-User Correlation:* Correlation across individuals is relatively complicated than correlation across context involving an individual. Also the correlation keeps varying across different groups of individuals. (ii) *Shared Context of Interest across Queries:* Multiple concurrent queries are likely to require the same context from the same individual. (iii) *Variable Processing Latencies:* Applications may need to be notified of collectively derived context within a specified time limit.

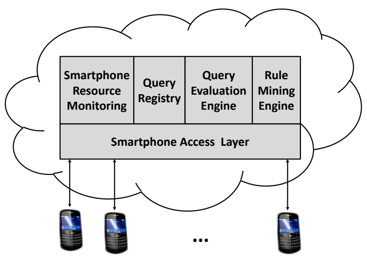

Fig. 1. The Overall Functional Architecture of *CloQue* .

To support such large scale energy-efficient evaluation of multi-person, continuous queries, we propose *CloQue*[1], a cloud-based framework. Applications submit their continuous collective-context based queries to the *CloQue* cloud engine, which then retrieves the required contextual states by dynamically tasking specific sensors on individual smartphones.

Our key contributions are: (i) applications can specify a probabilistic *confidence threshold* for each collective query. A key innovation is the use of two separate *confidence values* with each context predicate, which permits both deterministic and probabilistic queries to be short-circuited in a uniform way. (ii) We propose a novel metric called *normalized expected change in confidence (NECC)*, based on propagated confidence values, to dynamically determine a context evaluation sequence that balances acquisition cost, selectivity and coverage. (iii) By testing the performance of *CloQue* on a real-life dataset, we demonstrate that *CloQue* can achieve 50-60% energy reduction without sacrificing any query correctness.

II. The CLOQUE System Architecture

CloQue employs a client-server architecture, with a centralized query processing engine coordinating the sensing and context collection tasks across a large set of mobile devices. Figure 1 describes *CloQue's* functional architecture.

The *Smartphone Access Layer* handles all communications with the smartphones. The *Query Registry* allows different context-aware applications to issue continuous queries to the *CloQue* engine, and remove queries when they are no longer needed. The *Resource Monitor* tracks the resource levels at

[1]**Clo**ud-based **Que**ry Evaluation Framework, pronounced as 'cloak'

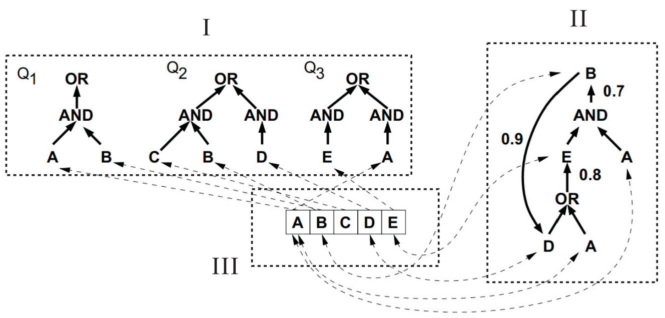

Fig. 2. Example of the 3 data structures used in query evaluation. The query set is I, rule set is labeled II, and distinct predicates list is III.

each smartphone. The *Rule Mining Engine* collects historical data about each individual and infers association rules from them, using standard ARM techniques, such as the *a priori* algorithm [3]. Finally, the *CloQue* Query Evaluation Engine (**QEE**) is the central coordinator that evaluates the continuous queries in the registry and sends the results to subscribing smartphones. In the rest of this paper, we focus principally on describing the QEE.

Context Query Representation: In *CloQue*, a query is a boolean combination of predicates in disjunctive normal form (DNF), modelled as a three-level tree, with the root node being the logical OR operator, the second level nodes representing the logical AND operators, and the leaf nodes representing the predicates. Figure 2 illustrates an example of a query. Each predicate *A* - *E* must be associated with at least one sensor potentially belonging to different users; however, two different predicates can operate on data from the same sensor. In the probabilistic setting of *CloQue*, each query is also associated with a user-specific *confidence threshold*, and the query is considered to be successfully evaluated when the probability or confidence that the query evaluates to true exceeds this threshold.

III. The *CloQue* Query Evaluation Engine

The goal of the *CloQue* QEE is to evaluate the set of queries in the Query Registry, while *minimizing the energy consumption of the set of smartphones*.

A. Probabilistic Query Evaluation

In *CloQue's* query evaluation, (i) each node in a query tree has two dynamically changing *confidence* values (between 0 and 1): the *true-confidence* denotes the current probability that the predicate is true, and *false-confidence* denotes vice-versa.(ii) association rules mined from historically observed cross-context correlation is used to update node confidence.(iii) the order of evaluation of queries is pre-computed to increase the likeliness of short-circuiting a query.

CloQue uses *association rules* to capture the inter-dependencies and correlation among multiple contexts. The head of a rule is a single predicate, while the body is a list of other predicates, such that head is true only if all the predicates in body are true. Different rules with the same head are treated as a logical OR relationship. A rule is associated with a *confidence* (the fraction of historical data where the head of the rule is true, given that the body is true).

Query Data Structure: The QEE maintains three data structures: the set of queries, the list of distinct predicates, and the set of rules as shown in Figure 2

The set of queries is represented as a forest of query trees. The two confidence values (*true-confidence* and *false-confidence*) for each context node are both initialized to zero. Evaluating a predicate at the smartphone will update the true-confidence and the false-confidence to 1 respectively depending on whether the predicate is true or false. The set of association rules is represented as a directed graph with two types of vertices: *Logical* vertices represent the logical AND/OR operators, while *predicate* vertices are identical to the predicate nodes in the query tree. Outgoing links from predicate vertices indicate query predicates.

B. Predicate Ordering for Query Evaluation

The predicate list data structure specifies an order to evaluate the predicates in order to minimize energy consumption via short-circuiting. The QEE evaluates the forest of queries in a bottom-up fashion, starting with the leaf nodes which are linked to the predicate list. These predicates are evaluated sequentially: the first predicate in the list is evaluated by querying the corresponding smartphone and retrieving the result, followed by *propagation of confidence values* (described in Section III-C) via the set of available association rules. As a result of such confidence propagation, if any query has been satisfied (the query confidence threshold met), QEE generates an alert to the application while proceeding to the next predicate.

QEE uses a dynamically re-ordered predicate list to reflect that confidence propagation can change the true-confidence and false-confidence values of predicates which is yet to be evaluated. *CloQue's* re-ordering algorithm is outlined in Alg. 1, and uses a new metric called *NECC* to balance several competing desires, preferring predicates that: *(i)* have a high probability of short-circuiting; *(ii)* incur less energy cost to evaluate, *(iii)* affect a larger number of queries (higher *coverage*); and *(iv)* will resolve the maximum amount of uncertainty about other un-evaluated predicates.

To capture objective (iv), we simulate the update propagation for the two hypothetical cases when a predicate z is true (t) and when the predicate is false (f). Suppose there are m internal nodes $\{q_1, q_2, ..., q_m\}$. The change in confidence assuming predicate $z = t$ is,

$$\Delta C\mid_{z=t} = \sum_{i=1}^{m} \Delta C_t(q_i)\mid_{z=t} + \Delta C_f(q_i)\mid_{z=t} \quad (1)$$

where $\Delta C_t(q_i)\mid_{z=t}$ is the change of an internal node's $true\text{-}confidence$ and $\Delta C_f(q_i)\mid_{z=t}$ is the change of an internal node's $false\text{-}confidence$. The change in confidence assuming that the predicate is false, $\Delta C\mid_{z=f}$, is computed similarly.

The *normalized expected change in confidence* (NECC) can be represented as:

$$NECC(z) = \frac{P(z)\Delta C\mid_{z=t} + P(\neg z)\Delta C\mid_{z=f}}{cost(z)} \quad (2)$$

where $P(z)$ (similarly $P(\neg z)$) denotes the probability that predicate z evaluates to be true (or false) and $cost(z)$ denotes

Algorithm 1 QUERY EVALUATION LOOP

Input: A set of queries $Q_k = \{q_1, q_2, ...q_m\}$, a set of rules R, a set of energy cost $Cost$, evaluation period ω

Output: Generate alerts for each query that is satisfied

```
1: Let $H$ be the priority heap for the predicate list by using Eqn. 2
2: for every $\omega$ seconds do
3:     for all predicate $h \in H$ do
4:         calculate the NECC for predicate $h$
5:     end for
6:     heapify($H$)
7:     while empty($H$) is false do
8:         $z \leftarrow$ extractMax($H$)
9:         $val(z) \leftarrow$ evaluate $z$ at phone
10:        UPDATE RULE CONFIDENCE($R, val(z)$)
11:        UPDATE QUERY CONFIDENCE($Q, val(z)$)
12:        for all $q_i \in Q_k$ that satisfied do
13:            generate alert for $q_i$
14:        end for
15:        for all predicate $h \in H$ do
16:            calculate the NECC for predicate $h$
17:        end for
18:        heapify(H)
19:    end while
20: end for
```

the cost (in terms of energy) of evaluating predicate z. After computing this value for all the un-evaluated predicates, the QEE next picks the one with the highest NECC value.

C. Confidence Propagation Using Rules.

After the evaluation of a predicate at the smartphone, the *CloQue* query engine updates the confidence values in the query forest using the association rules. The query engine first propagates the updated confidence values through the rule graph (note that these updates can change the confidence values of other predicates as well), and then propagates the updated confidence values up the query trees.

Confidence propagation is performed independently for the true-confidence and the false-confidence values. Let $C_t(u)$ and $C_f(u)$ denote the true-confidence and the false-confidence of a node u in either of the three data structures. The update logic is based on the intuition that the true-confidence of an OR-node is the maximum confidence of the true-confidence of its predecessors and the true-confidence of an AND-node is the minimum confidence of the true-confidence of its predecessors. For the rule graph where a predicate node can have incoming edges associated with a rule-confidence, the true-confidence of a predicate node given that its predecessor's true-confidence has been updated is the rule-confidence multiplied by the predecessor's true-confidence. The following update equation summarizes the bottom-up update logic for the true-confidence value of node v given each successor node u of node,

$$C_t(u)^{(n+1)} = \begin{cases} \max\{C_t(u)^{(n)}, C_t(v)^{(n)}\} & \text{if } u \text{ is an OR} \\ \min_{\omega \in Pred(u)} C_t(\omega)^{(n)} & \text{if } u \text{ is an AND} \\ \max\{C_t(u)^{(n)}, C_t(v,u) \cdot C_t(v)^{(n)}\} & \text{if } u \text{ is a predicate} \end{cases} \quad (3)$$

where the superscript n and $n+1$ denote the time before and after one application of the update equation. The term $C_f(v,u)$ denotes the confidence of an association rule. Note that for the rule graph, the update propagation only updates the true-confidence, as association rules only apply when its body is true.

Algorithm 2 PARTITION THE QUERIES

Input: A set of queries $Q = \{q_1, q_2, ...q_m\}$, a latency threshold l
Output: p partitions $Q^p = \{Q_1, ..., Q_k\}$

```
1: for all $q_i \in Q$ do
2:     Caculate the estimated evaluation time $t_{qi}$ of $q_i$
3: end for
4: sort $q_i \in Q$ in desending order by $t_{qi}$
5: for all $Q_i \in Q^p$ do
6:     while pop $q_i$ from $Q$ do
7:         $Q_i$'s estimated evaluation time $t_{Q_i} \leftarrow \sum_{q_j \in Q_i} t_{qj}$
8:         if $t_{Q_i} + t_{q_i} < l$ then
9:             add $q_i$ to $Q_i$
10:        end if
11:    end while
12: end for
13: sleep until being awaked
```

IV. CLOQUE: Implementation and Evaluation

We have implemented a working prototype of *CloQue*, with the Query Evaluation Engine implemented in a perl-based engine and evaluated it using a large-scale dataset: the Reality Mining dataset [5] which was replayed appropriately to the Query Evaluation Engine.

A. Queries and Energy Profiles Used

To test different variants, we designed queries to mimic three different scenarios of every day events of interest in workplace settings: *a)* **Interruptibility**: –both individual (e.g., *"Bob is at work and is not using his phone"*) and group-level (e.g., *"Bob and Jack are both at work and are not using their phones"*; *b)* **Group Semantics:** *"Bob, Jack, and Ross are together at the Cafeteria"*; and *c)* **Proximity Alerts:** e.g., *"Bob and Jack are near each other in any building"*.

We created 3 different query sets (one for each scenario listed above) for our dataset. Each query set used trace data from at least 20 different smartphone users. A total of 63 unique predicates in the dataset were used. We used the Monsoon Power Monitor [1] to measure the power consumption of a Samsung Galaxy S3 phone [2] running on Android version 4.0.3 to get the energy consumption values of the sensors.

B. Four Implementations Used for Evaluation

- ***Naive*** every sensor specified in a query is evaluated—i.e., the evaluation of a query set is not complete until all the predicates in each of the queries has been evaluated. The choice of *Naive* where no collaboration takes place is similar to the baseline chosen by [6]
- ***Short-Circuit*** queries are evaluated in order until a result is deterministically known and then processing is short-circuited. However, the order of query processing is fixed in a FIFO order. This is similar to the approach in [9].
- *CloQue$_{NoRules}$* is a variant of *CloQue* that intelligently reorders queries but does not use the association rules and confidence propagation mechanisms described in Section III-A.
- *CloQue$_{Full}$* is the full implementation of *CloQue* as described in Section III. The main difference from

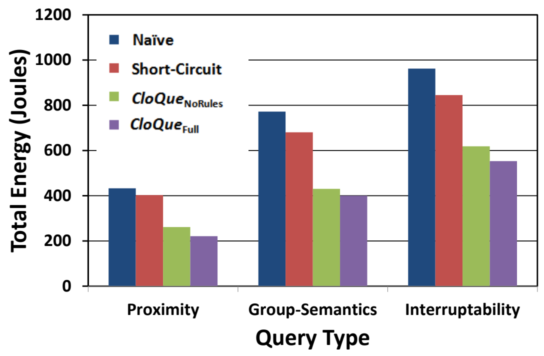

Fig. 3. Energy Consumption of the Four Variants in Reality Mining dataset

Dataset	Prox.	Grp-Semantics	Interuptability
Reality Mining	18.27	7.52	11.84

TABLE I. IMPROVEMENT IN ENERGY SAVINGS (%) BETWEEN $CloQue_{Full}$ AND $CloQue_{NoRules}$

$CloQue_{NoRules}$ is that the full version of *CloQue* uses the association rules and confidence propagation mechanisms to trade-off a little accuracy for extra energy savings.

C. Results: Base Evaluation

Figure 3 shows the total energy consumption for the Reality Mining Dataset. The result shows that, relative to *Naive* and *Short-Circuit*, the 100% accurate version of *CloQue* ($CloQue_{NoRules}$) reduces the total energy consumption by about 50% with the full version of *CloQue* ($CloQue_{Full}$) doing even better than $CloQue_{NoRules}$.

Table I shows, in more detail, the benefits of turning on the association rule engine in *CloQue*. In particular, we can save 18.27% and 11.84% more energy for proximity and interruptibility type queries while saving 7.52% more energy for group-semantics based queries. The accuracy obtained by $CloQue_{Full}$ is also between 95 to 96% when the confidence is larger than 90%. Thus the full version of *CloQue* provides up to 18.27% energy savings (depending on the type of query) for a modest 4% accuracy loss.

The energy improvements are not consistent across all the smartphone users. However, the energy consumed at each phone by $CloQue_{NoRules}$ and $CloQue_{Full}$ is significantly lower than the other two implementations – with $CloQue_{Full}$ consuming about 12.08% less energy per phone, on average, than $CloQue_{NoRules}$.

D. Results: Sensitivity Analysis

We investigated the effect of changing confidence thresholds of the association rule engine.

Confidence Thresholds: Table II shows the effect of changing the confidence values of *CloQue's* (using the $CloQue_{Full}$ variant) association rule engine. In the Social Evolution dataset, we observe that reducing the confidence from 95% to 50% results in a 33.3% reduction in energy consumption but at the cost of an almost14.5% reduction in accuracy. Overall, we found, that for these two datasets, confidence of 90% (at 10% support) gave the best trade-off between energy consumption and accuracy.

Conf. (%)	50	60	70	85	90	95
Accu. (%)	75.2	85.2	90.8	93.7	95.2	96.0
∑Energy (J)	413.3	455.3	485.9	506.8	515.0	523.5

TABLE II. EFFECT OF CHANGING CONFIDENCE LEVELS

V. CONCLUSION

We presented *CloQue*, a cloud-based query evaluation system for optimizing the overall energy consumption of group-based queries across multiple smartphones. *CloQue* achieves energy savings by exploiting: (i) variable acquisition cost of different sensors and (ii) correlation among different phones arising from shared human activity context. Our experiments using real traces from a large real-world dataset shows that *CloQue* can reduce overall energy consumption by up to 60% with only a 4% loss in accuracy. In our future work, we plan to deploy the *CloQue* system on real phones and users, and evaluate *CloQue* in even more realistic online settings. Presently we have not addressed the challenges of latency-constrained optimization, which we plan to address in the future by (i) varying the evaluation period of queries to determine the trade-off between energy saving and accuracy and (ii) partitioning the queries into multiple smaller partitions and evaluating the various partitions in parallel and independently. Independent evaluation of partitions can lead to redundant evaluation of certain predicates which we intend to address by maintaining a cache which is accessible to all partitions.

REFERENCES

[1] Monsoon. http://www.msoon.com/LabEquipment/PowerMonitor/.

[2] Samsung galaxy s3. http://www.samsung.com/global/galaxys3/.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, Santiago, Chile, Sept. 1994.

[4] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, Toronto, Canada, 2004.

[5] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal & Ubiquitous Computing*, 10(4):255–268, Mar. 2006.

[6] Y. Lee, Y. Ju, C. Min, S. Kang, I. Hwang, and J. Song. Comon: cooperative ambience monitoring platform with continuity and benefit awareness. In *MobiSys*, Low Wood Bay, UK, June 2012.

[7] L. Lim, A. Misra, and T. Mo. Adaptive data acquisition strategies for energy-efficient, smartphone-based, continuous processing of sensor streams. *Distributed and Parallel Databases*, 31(2):321–351, May 2012.

[8] S. Nath. Ace: exploiting correlation for energy-efficient and continuous context sensing. In *MobiSys*, Low Wood Bay, UK, June 2012.

[9] J. Pei, M. Hua, Y. Tao, and X. Lin. Query answering techniques on uncertain and probabilistic data: tutorial summary. In *SIGMOD Conference*, Vancouver, Canada, June 2008.