Towards Using Data Aggregation Techniques in Ubiquitous Computing Environments

Faraz Rasheed, Young-Koo Lee, Sungyoung Lee

Kyung Hee University, Suwon
449-701, Republic of Korea
faraz@oslab.khu.ac.kr, {yklee, sylee}@khu.ac.kr

Abstract

Ubiquitous Computing envisions the era of computing where pro-active computing services are available to all any where, any time. In order to provide such pervasive services, a ubiquitous system collects a large amount of information continuously from its physical and computational environment. Managing and manipulating this information optimally is one of the most important aspects of ubiquitous computing system. We have proposed formal schemes for the storage and management of this information using the summarization techniques. We prepare aggregates or summary of information present in the repository and try only to use these summaries and aggregates most of the time for query processing. This approach not only saves the storage space but also results in efficient distributed and mobile data processing.[1]

1. Introduction

Pervasive Computing envisions [5] a computing environment where different devices are immersed in the environment invisibly and providing pro-active context aware services to its users. In order to provide such services, the system needs to be aware of target user's context like current activity, users involved, location, time, environment properties (temperature, noise level, luminous intensity), computational environment (current bandwidth) etc

Usually a ubiquitous system has to collect a large amount of information continuously from sensors and other devices. Such a huge amount of information requires special consideration for its storage, retrieval, and reasoning. The efficient manipulation and processing of such information is the key to realize the dream of pervasive computing.

We have proposed [2] the formal techniques for the use of knowledge summarization in ubiquitous computing environment. We first filter out the un-necessary information (through information filtering) and keep the remaining information in summarized form that represents the whole data resulting in the consumption of less storage space.

The rest of the paper is organized as follows: Section 2 introduces the concept of summarization and motivation behind it, Section 3 presents some formal techniques that can be used, Section 4 shows the architecture of summarization modules and query translation required in the system. Finally in Section 5, we present some related work and conclude the paper in Section 6.

2. Knowledge Summarization

Context aware systems define context as any information that is useful in describing the current situation or context of user. For pervasive computing environment, hence, the context could be current temperature, activity, location, noise level, users involved, current time, user schedule, etc. Our Summarization techniques try to identify the useful context information and keep the summarized context information in the knowledge repository for further use.

Context Summarization (CS) is a method of representing raw context information into summarized information so that it takes relatively less storage space and can successfully answer the queries for complete information with acceptable degree of confidence.

Such a compact representation of information reduces required storage space. Since Ubiquitous computing systems are usually distributed and contains a lot of mobile devices, hence using summarization, fewer amounts of data is required to be transferred over the network. These systems have a large amount of information so querying over the collection of data is also

[1] This research is supported by the Driving Force Project for the next generation of Gyeonggi Provincial Government in Republic of Korea
Professor Sungyoung Lee (sylee@khu.ac.kr) is the corresponding author for this paper

inefficient. For example, if we are saving the temperature information after every five minutes then after a month, querying for the temperature value at certain time would consume considerable processing. But if the same query is applied on the aggregate representation, it would take significantly less time and processing. Although the result of queries over aggregate or summarized information might not be 100% accurate, but since most of the time this information is required for system's internal use (for inference making, machine learning, knowledge reasoning), we can work out with approximate values with certain degree of confidence. Finally, summarized information is also useful for faster inference making, user preference learning, data mining, machine learning and knowledge reasoning.

3. Few Summarization Techniques

In this section, we will identify few techniques for the summarization or aggregation of information. All of these techniques are suitable for a different set of information and are specific to the nature of pervasive applications and how they are using it.

3.1. Aggregation

The simplest and most commonly used form of summarization is aggregation where we represent the collection of information in fewer values and use this aggregate information to reply the queries later. The simplest form of aggregation is average or mean. But according to the context type and application requirement, more suitable forms can be used like considering variance, standard deviation and rate of change. For example, if a temperature sensor is giving temperature values after every 5 minutes, then we can repeatedly calculate and generate aggregates as shown in Table 1.

Table 1. Aggregate values of temperature

Date	Period	Avg. Temp	Min. Temp	Max. Temp
09/18	Morning	16	13	18
09/18	Afternoon	20	18	23
09/18	Evening	18	17	20
09/19	Morning	17	14	18

3.2. Pattern Identification

Context information can be summarized by identifying general patterns and later answering approximately to the queries using these patterns. For example, location information can be summarized using this technique. Let's say, we are storing the location of different users after every 15 minutes, then we can identify user location patterns out of this data. Suppose we have location information of users as presented in Table 2.

Table 2. Location Information for users

Time	User	Room
09:05	1	1
09:02	2	1
09:02	3	1
10:08	1	2
10:37	5	2
10:59	6	3
11:26	3	3
11:44	3	3
…		

The above location information can be represented in patterns as in Table 3.

Table 3. Patterns for location of users

Time Period		User	Room	Probability
From	To			
09:00	12:00	1	1	0.76
13:00	17:00	1	1	0.83
09:00	12:00	2	2	0.67
13:00	17:00	2	1	0.89
14:00	19:00	4	3	0.36

3.3. Other Techniques

Other techniques like categorization, feature extraction, drift calculation and generalization can also be used for summarization [2].

4. Architecture & Working

As we mentioned in our earlier paper [2] that there are two types of summarization Active (or Instantaneous) and Passive (or Delayed) summarization. Here we will only discuss the delayed summarization (summarization performed on the information repository) because of the complexity involved in it. Figure 1 presents the architecture for our summarization module

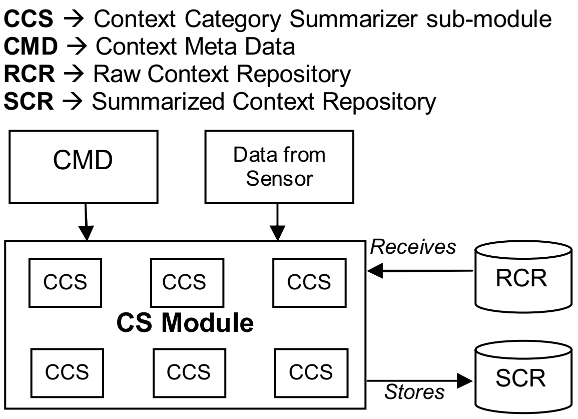
CCS → Context Category Summarizer sub-module
CMD → Context Meta Data
RCR → Raw Context Repository
SCR → Summarized Context Repository
CMD
Data from Sensor
CCS
CS Module
Receives
RCR
Stores
SCR

Figure 1. Context Summarization Module

Each category of context is dealt by corresponding CCS using one of the summarization techniques mentioned earlier. A new context type is introduced in the system with metadata (CMD) describing its context type so that appropriate summarization techniques can be applied. The metadata also includes key fields on which to apply summarization, the representation of summarized information (e.g., to include a particular field in the summarized information), the source and target location for retrieval and storage of data and the default summarization strength; the degree to which summarization is to be performed.

We are using our middleware CAMUS [3] to apply the context summarization. The interaction of summarization module with other components of middleware is presented in Figure 2.

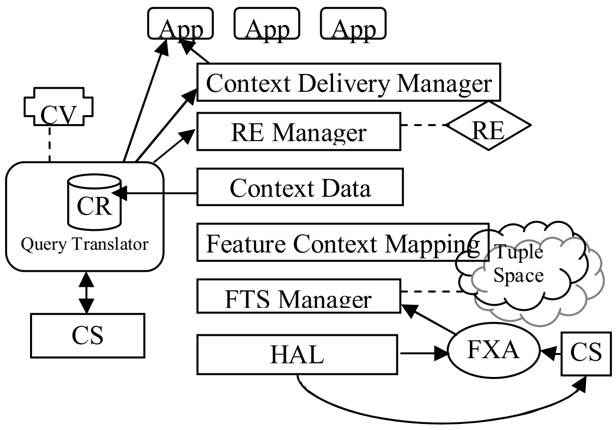

Figure 2. Interaction with other middleware modules

First we extract features (unified representation of sensory data) through our Feature Extraction Agents (FXA) and store all these features in Feature Tuple Space (FTS) which is an in memory repository of current context or the latest information received from sensors. As a new instance of information is inserted in FTS, the older one is transferred to the Context Repository (CR) represented using ontology in OWL through Feature-Context Mapping Layer. From then, all the middleware modules (reasoning engines, middleware services) and application access this information from the context repository. As data is stored in the repository, we summarize this information timely and store back to repository. One approach (used in case of temperature, humidity, etc) the raw information is removed from the repository and only the summaries or aggregates are used to answer queries. Another approach is to keep multiple summaries of different strength are kept and used to reply the query with appropriate confident values. A hybrid approach can also be used in which both summaries and raw information are kept; specific or precise queries are answered from raw data while the general queries are answered through summarized information.

4.1. Query Translation

Context Summarization modules change the context repository and form data units with different schema than the original one. How can context consumers cater with this? How do they know whether particular information is in summarized state or it is still in raw form? As in Figure 2, there is a special module called Query Translation (QT) which encapsulates context repository (CR). All other modules (CS, Reasoning Engine, Applications, etc) interact with repository through QT. Query Translator makes all the access to CR transparent, i.e., even the modules and applications are not required to be aware of summarization process. It keeps track of partition of summarized and raw data and directs the access to these accordingly by intercepting each and every access to CR. If the required data has been used in the summarization, it directs the queries to the summarized data repository. The results produced due to QT are not 100% accurate; hence it also returns a confidence value with each query result. Further, a query may also specify the minimum degree of confidence for the required results.

4.2. Context Category Summarizer (CCS)

Each category of context is summarized by a particular Context Category Summarizer (CCS); hence there is a different CCS for aggregation, pattern identification, etc based information. For example, temperature, available network bandwidth and noise level can be summarized using aggregation based CCS. Each CCS instance contains

(a) summarization algorithm,
(b) general parameters (key field, required fields, etc),
(c) specific parameters (source & target data source, summarization strength, time interval for repeated

invocation of summarization),
(d) query translator for summarized information

Context Summarization Manager also maintains a list of context information used by different CCS for summarization, an example for such a table is present in Table 3.

Table 3. List of context information summarized by different CCS

CCS_ ID	CCS_ Instance_ID	Context_Type_ ID	Last_Updated
017	1	1 (temperature)	09/19/05 05:42
017	2	4 (light intensity)	09/19/05 11:37
019	1	21 (location_A)	09/19/05 17:16

Using this list, a Query Translation Manager can identify whether a particular context information type is summarized and also if the required data has been in summarized or it is still in raw format. Moreover, if the required information is in summarized state, then which CCS's QT should be invoked to get the query result? The general process flow of query processing is presented in Figure 3.

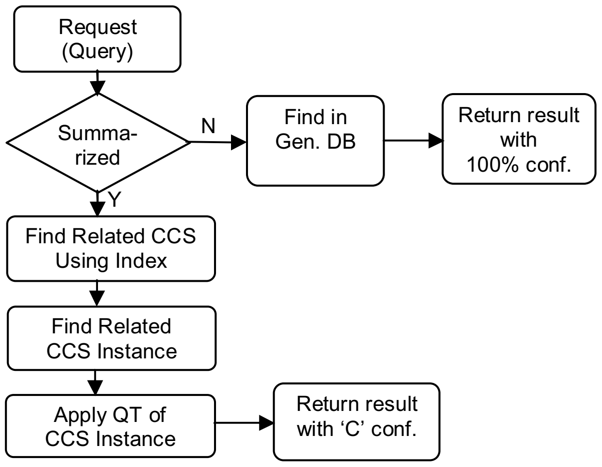
Request (Query)
Summa- rized
N
Y
Find in Gen. DB
Return result with 100% conf.
Find Related CCS Using Index
Find Related CCS Instance
Apply QT of CCS Instance
Return result with 'C' conf.

Figure 3. General Process Flow of Query Processing

5. Related Work

Several existing systems support techniques like feature extraction and generalization [4][6] but we want to formally make summarization as part of the ubiquitous system's data management. Our idea is to generate summaries so that later we don't need the raw data any more and can reply to most of the queries with this summarized information with acceptable degree of confidence. In DBMS, data mining and data ware housing [1] use the concept of histogram and multidimensional views of database and work on the aggregate, consolidated data instead of raw data to support the higher level decision making and to identify the hidden patterns in the data. The goal of data mining and OLAP is somewhat similar but we want to transform the raw context to summarized form taking less storage space and provide improved and efficient reasoning and machine learning. Aggregate data analysis has also been discussed in DBMS for quite sometime which is also helpful for this kind of work.

6. Future Work & Conclusion

The foremost important issue is the performance cost and the selection of time interval for the invocation of summarization. Synchronization of the different CCS modules is also an important consideration. If too many CCS modules start performing summarization then the overall system performance might degrade. Also there might be some queries for the data that is currently being summarized; we are also working on implementing the appropriate locking mechanism. Another interesting future work is to implement hierarchical summarization with different summarization strength and which allows inter-module negotiation [7] for required summarization strength and confidence values for queries. For our future work, we also want to use the concept present in [7] for summarization strength negotiation.

7. References

[1] Alex Berson , Stephen J. Smith, Data Warehousing, Data Mining, and OLAP, McGraw-Hill, Inc., 1997

[2] Faraz Rasheed, et al: Context Summarization and Garbage Collecting Context; Computational Science and Its Applications – ICCSA 2005: International Conference, Singapore, May 9-12, 2005, Proceedings, Part II.

[3] Hung Q. Ngo et al: Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework. *In Proceedings of the EUC 2004*: 672-681.

[4] Jason I. Hong, James A. Landay. Support for location: An architecture for privacy-sensitive ubiquitous computing, *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, June 2004

[5] M. Weiser, The computer for the 21st century. *ACM SIGMOBILE 1999 Review.*

[6] Mike Spreitzer, Marvin Theimer, Providing location information in a ubiquitous computing environment, *ACM SIGOPS Operating Systems Review , Proceedings of the fourteenth ACM symposium on Operating systems principles* Dec 1993, Volume 27 Issue 5

[7] Khedr, M. Karmouch, A: Negotiating context information in context aware systems. *IEEE Intelligent Systems* Dec 2004