Applying Context Summarization Techniques in Pervasive Computing Systems

Faraz Rasheed, Young-Koo Lee, Sungyoung Lee
Dept. of Computer Engg. Kyung Hee University
Suwon, 449-701, Republic of Korea
faraz@oslab.khu.ac.kr, {yklee, sylee }@khu.ac.kr

Abstract

Pervasive Computing Systems are connected with a number of sensors. The system receives a lot of information continuously from sensors to sense its physical and computational environment. Knowledge Management in such systems imposes several challenges such as ubiquity and distributed knowledge acquisition. As the amount of data grows, the query processing efficiency is reduced. We proposed context summarization and use of aggregate information in such systems. The idea is to summarize the received information and try to use this compact and reduced sized information as much as possible to answer user queries. In this paper, we are going to describe some of the techniques that can be employed for such a summarization process.[1]

1. Introduction

Ubiquitous or Pervasive computing envisions [1] an environment where computer systems and devices are spread through out the environment assisting humans to perform their routine tasks proactively and seamlessly. Thus a typical ubiquitous environment contains a number of sensors, computing and networking devices along with complex software sub-systems. Several ubiquitous computing projects [2] [3] [4] [5] are under development.

Context Awareness is the prime technique to make pervasive computing seamless and proactive. In order to provide services proactively, the system needs to be aware of the users and environmental context. But what context actually is? We take context as any 'implicit situational understanding' of the world considered and all the information which define the considered situation.

One distinguishing aspect of ubiquitous systems is the amount of context information (or context) it receives from its physical and computational surroundings. Using this huge amount of context information efficiently, reasoning over it, making inferences about user activity, context aware application adaptation, machine learning and data mining. Thus context management is the foremost important part of any ubiquitous system [7]

We approach the context management from a different angle and question about keeping the raw context in context repository. We propose [6] [14] that instead of (or in addition to) keeping the raw context in the repository; it makes sense to summarize the available context and keep this summarized and inferred context in the context repository. We believe that such a summarization would result in saving the available storage space and also provide the better query processing, reasoning, machine learning and identifying the future intention of user.

The rest of the paper is organized as first we will define the current issues, present and motivate the solution with the context summarization (Section 2). In section 3 we will present few techniques to achieve the context summarization. In section 4, we will present our current architecture that how we manage the context summarization module. Later we will present (Section 5) some research issues and challenges in the field. We mention about some related work (Section 6). We will conclude the paper with future work in section 7.

2. Context Summarization

The context information comes in a continuous stream with each sensor emitting the data regularly (at least during some interested activity). We are heading towards flood of context data! Such a huge amount of data requires proper management and formal management techniques. At this point, we need to answer what to do with such a huge amount of data? Do we need to store all of this data? More importantly do we really need such a large amount of data? Figure 1 presents the data injection in typical ubiquitous computing environments.

---

[1] This work is supported in parts by Korean Ministry of Information and Communication (MIC) and Information Technology Research Center (ITRC)
Prof. Sungyoung Lee (sylee@khu.ac.kr) is the corresponding author for this paper

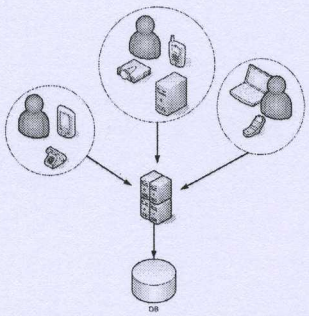

Figure 1. Data injection in pervasive systems

Several context items sensed from the physical and computational environment are consumed inside the middleware system to reason for higher level context, activity recognition, for data mining and proactive pervasive services. Hence, we can afford to change the internal representation of data.

Context Summarization [6] [14] is the process of representation of existing information in such a way that it consumes relatively less storage space and can still answer most of the user queries with required confidence level. For example, consider a temperature sensor emitting the temperature value after every 5 minutes. We can simply store this as it is coming. Table 1 demonstrates this case.

Time	Temp.
12:05	23 °C
12:10	21 °C
...	

Table 1. Temperature values after every 5 minute

Using Context Summarization, we can summarize this information and group on the daily basis (Table 2)

Date	Period	Avg. Temp	Max. Temp	Min. Temp
12/01	Morning	5 °C	8 °C	3 °C
12/01	Afternoon	10 °C	14 °C	8 °C
12/01	Evening	9 °C	11 °C	7 °C
12/01	Early Night	7 °C	8 °C	5 °C
12/01	Late Night	4 °C	5 °C	2 °C
...				

Table 2. Temperature values stored for different periods of day

Using the same concept, location and computational environment context (like network bandwidth and processor load at a particular time) can also be summarized. The benefits of performing the context summarization include reduced storage space. Such a compaction of data repository will save significant amount of storage space which will result in the faster query execution and data retrieval. It will also make the data migration in distributed environment more efficient.

3. Techniques of Context Summarization

3.1. Aggregation

In aggregation, the history of context information is aggregated to generate compact and consolidated context. Numerical context types like temperature, light intensity, pressure, humidity, available network bandwidth and state of current system resources can be summarized using this technique. In previous section, we have demonstrated how this technique can be used to summarize numerical context information history.

The simplest form of aggregation is using averages, standard deviation and then providing the nearest value from the available interval aggregate value.

Another method is storing the values at the end of interval and storing a time series graph values at elongated period compared to the original interval. For example, if by default the values are received after every 5 min., the time series graph will record values after every 30 minutes. Figure 2 demonstrates one such example. Here, we can use curve fitting methods to find more appropriate curve for the available data points and using the curve, we can answer the queries for data points that are not available explicitly in this curve (e.g., for time 11:47)

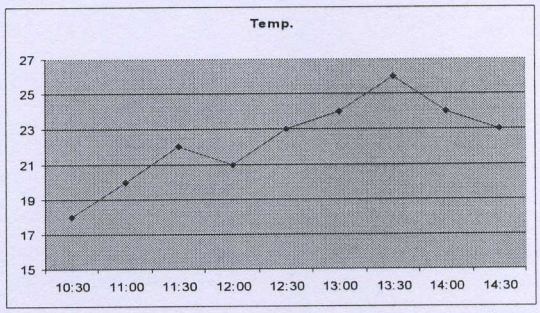

Figure 2. Data points stored for sparse time values

Similarly, we can also use probability density functions (PDF) for storing the values against time with certain probability.

We are using aggregation not only to summarize the numerical contexts but also for the summarization of Location-Map (or radio map) for wifi based location awareness data. The summarization techniques are being applied to training data for neural networks which grows in multiple of 10,000 for the location map of few floors!

We are getting some encouraging results with good amount of storage space reduction and query results with more than 70% accuracy.

3.2. Categorization

This technique categorizes context entities and summarizes these categories. For example, context information like user profile and device profile can be categorized to form user or device groups having some similar properties. In this way, we can identify the activities and features of a particular group or category like we can track the network bandwidth utilization by some particular user group (say doctors) or by some particular device group (say PDAs) during office hours.

Categorization is delayed and usually static type of context summarization, i.e., it is not performed and changed frequently. Categorization can be performed at system startup by some human or the system can learn itself and define categories as it is executed for elongated period of time. In any case, the categorization supports Machine Learning and higher level reasoning.

3.3. Context Extraction

In Context Extraction, useful and interested context is extracted from continuous context streams such as audio and video streams. For example, Context Extraction can be applied to video stream received from video sensors like Camera, Webcam to extract features like pixel percent change, pixel change variance, picture motion pattern (such as stable, regular, irregular), luminous intensity, etc. In the similar way, audio context can also be summarized.

It may discard the original (or raw) context even before it being stored in the repository; hence resulting in only storing the extracted features and not the original data. It results in saving a lot of storage space but may take considerable time in doing so. Some ubiquitous computing projects [13], including our project CAMUS [5], have been using this technique for some time.

3.4. Activity Sequencing

Context Extraction can also be implemented by recording activities over time for a location and its participants. It means that instead of storing the sparse context isolated values, data is stored in such a form that it narrates a spatio-temporal sequence of activities along with its participants. In this process some unnecessary, redundant and less useful information can be removed from the details. The result of query then can be produced by analyzing the activity and more importantly in context of activity. For example, if the system has the record that Bob had a party in his home last Saturday from 10 PM to 2 AM and users X, Y and Z were the participants. Then the system can answer higher level queries like what was user X doing on last Saturday, who were along with her, what sort of party that was and how long did she stay there; when it doesn't explicitly have this information about user X. Hence, there is no need for saving the explicit information for users X, Y, and Z activities.

3.5. Pattern Identification

Context information can be summarized by identifying existing patterns in the context repository or history of activities. Consider the location context history stored in the context repository as depicted in Table 3

Time	User	Room
09:05	1	1
09:02	2	1
09:02	3	1
10:08	1	2
10:37	5	2
...		

Table 3. Location Context History of Users and Rooms

Using pattern identification, a system may deduce the pattern of user's location during week days. See Table 4.

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
...				

Table 4. Pattern Identification for User Location

In the similar way, system can find the pattern of room occupants during various time periods. Using categorization along with pattern identification, system may also infer which user group (doctors, programmers, operators) occupies which room at different time periods. Pattern Identification is resource intensive but it results in reducing considerable amount of storage space and also supports higher level inference making, machine learning and in predicting future intentions of a user or expected behavior of a device in the current situation.

3.6. Generalization

In generalization, we map various ranges of context to a general higher level context. For example, we can map

the raw temperature and network bandwidth to general concepts as presented in Table 5 and 6 respectively.

Temp. Range (°C)	Generalized Weather
20 ~ 30	Hot
10 ~ 19	Moderate
0 ~ 9	Cold

Table 5. Generalization of Temp. Range

Network Bandwidth Range Available	Generalized Network Traffic
1 mbps or more	Mostly free
500 kbps ~ 1 mbps	Moderately used
Less than 500 kbps	Busy

Table 6. Generalization of Network Bandwidth

Actually generalization is an instantaneous kind of summarization performed between the sensor and middleware. This real time summarization only supplies the values to middleware when there is the difference of context general state. This type of summarization will reduce the processing burden from the middleware in addition to saving the storage space used We can combine generalization with other techniques like Aggregation and Pattern Identification to further optimize the system.

4. Architecture & Working

As we mentioned in our earlier papers [6] [14], that there are two types of summarization Active (or Instantaneous) and Passive (or Delayed). Here we will only discuss the delayed summarization (summarization performed on the information repository). Figure 3 presents our architecture

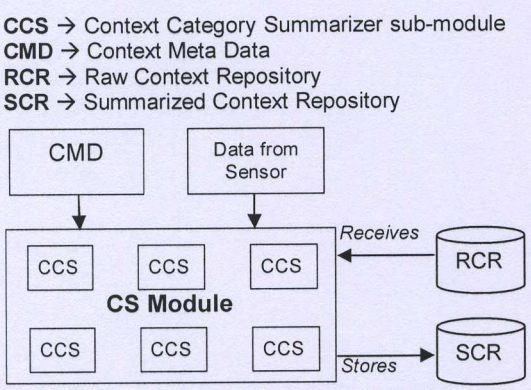

Figure 3. Context Summarization Module

Each category of context is dealt by corresponding Context Category Summarizer (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 [5] to apply the context summarization. The interaction of summarization module with other components is presented in Figure 4.

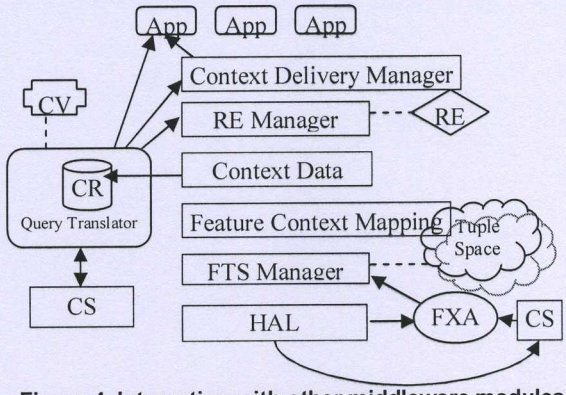

Figure 4. Interaction with other middleware modules

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. 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 confidence 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. It 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. If the required data has been used in the summarization, it directs the queries to the summarized 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 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

Figure 5 presents the interaction of sub-modules of CS engine. CCS Registry registers all the CCS and runs various CCS instances run in parallel separate threads. The mediators provide access to various storage media.

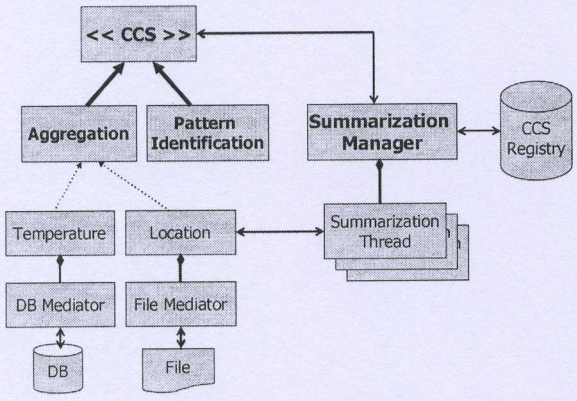

Figure 5. Interaction of core summarization modules

CS Manager also maintains a list of context information used by different CCS for summarization. See Table 7.

CCS_ ID	CCS_ InstanceID	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

Table 7. List of context used by different CCS

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 for query result?

5. Issues & Challenges

Context Summarization (CS) has its own unique research issues and challenges both at conceptual and implementation level. Here we identify several such issues and wherever possible identify few applicable solutions.

5.1. Performance Overhead

Perhaps the foremost concern to apply CS techniques is the performance cost. The overall system should not be ceased or hung-up during the execution of CS modules and the resources like context repository should not be locked for noticeable period of time. We solve this problem by using the context aware CCS invocations, i.e., summarizers (CCS) are executed when system has relatively less processing load.

5.2. Security & Risks

Security is the most questionable part of today's computing systems. The CS modules operate totally inside middleware and directly access & modify the context information which is the most valued asset of any ubiquitous system. Hence, the components and modules must be administered and validated carefully.

About the risks involved, firstly Context Summarization (CS) results in some data and precision loss. Failing to compensate this precision loss may result in decreasing the performance and throughput of the system. Secondly, improper Context Summarization may make the reasoning and machine learning even more difficult, complicated, inefficient, incorrect and misleading instead of improving it. Finally, several modules of middleware and application might be accessing the Context Repository at the same time. A sudden modification by CS might be unexpected for these modules and may make them produce unexpected results which must be avoided.

6. Related Work

In our previous papers [6][14], we presented the idea and motivation for Context Summarization (CS) and Context Garbage Collection (CGC). Several existing ubiquitous computing systems support features like feature extraction, generalization [8] [5] but we want to formally make summarization as part of the ubiquitous system's data management.

In Database Management Systems (DBMS), there are techniques that deal with similar problems. Data mining [9] and data ware housing [10] use the concept of histogram [11] and multidimensional views of database and work on the aggregate, consolidated data instead of raw data. Online Analytical Processing (OLAP) and data mining is not done on the actual data but on the historical, consolidated and aggregate data while we are performing the context summarization on the actual context. 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. Researchers in DBMS have also analyzed the time series data streams for very large databases [12]. Here, they analyze the data coming in continuous streams with time. They have proposed solutions on how to manage, represent and store the time series data streams. Aggregate data analysis [15] has also been discussed in DBMS for quite sometime which is also helpful for this kind of work.

7. Future Work & Conclusion

Synchronization of the different CCS modules is 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 working on implementing the appropriate locking mechanism. Another interesting work is to implement hierarchical summarization with different summarization strength which allows inter-module negotiation [16] for required summarization strength and confidence values. For our future work, we also want to use the concept present in [16] for summarization strength negotiation

8. References

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

[2] Dey, A.K., et al.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Anchor article of *a special issue on Context-Aware Computing, Human-Computer Interaction (HCI) Journal*, Vol. 16. (2001)

[3] Chen Harry, Tim Finin, and Anupam Joshi: An Intelligent Broker for Context-Aware Systems. In: *Ubicomp 2003*, Seattle, Washington

[4] Manuel Román et al.: Gaia: A Middleware Infrastructure to Enable Active Spaces, In *IEEE Pervasive Computing*, Oct-Dec 2002

[5] Hung Q. Ngo, Anjum Shehzad, Saad Liaquat, Maria Riaz, Sungyoung Lee: Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework. *EUC 2004*: 672-681

[6] Faraz Rasheed, et al: Context Summarization & Garbage Collecting Context, , UWSI 2005, In the proceedings of ICCSA 2005, published by Springer Verlag

[7] Michael J. Franklin, Challenges in Ubiquitous Data Management. . Informatics: 10 Years Back, 10 Years Ahead, LNCS #2000, R. Wilhiem (ed)., Springer-Verlag 2001

[8] 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

[9] Alex Berson , Stephen J. Smith, Data Warehousing, Data Mining, and OLAP, McGraw-Hill, Inc., New York, NY, 1997

[10] Inmon, W.H., Building the Data Warehouse. John Wiley, 1992

[11] D. Barbara et al., The New Jersey Data Reduction Report, Bulletin of the *IEEE Technical Committee on Data Engineering* December 1997 Vol. 20

[12] Lin Qiao et al: Data streams and time-series: RHist: adaptive summarization over continuous data streams, *Proceedings of the eleventh international conference on Information and knowledge management*, Nov 2002

[13] Moore, D., I. Essa, and M. Hayes, Exploiting Human Actions and Object Context for Recognition Tasks, In Proceedings of IEEE International Conference on Computer Vision 1999 (ICCV'99), Corfu, Greece, March 1999

[14] Faraz Rasheed, et al: Towards using data aggregation techniques in ubiquitous computing environment, Accepted for publication in PerWare 2006, In the proceedings of PerCom 2006, published by IEEE

[15] Joseph M. Hellerstein, Peter J. Haas, Helen J. Wang: Online Aggregation. SIGMOD Conference 1997: 171-182

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