Towards Real-Time Data Integration and Analysis for Embedded Devices

Michael Soffner, Norbert Siegmund, Mario Pukall, Veit Köppen
Otto-von-Guericke University of Magdeburg
{soffner, nsiegmun, pukall, vkoeppen}@ovgu.de

In complex systems, e.g., in logistic hubs, cars or factories, there is a need for real time decision support. In present approaches the transfer and storage process and subsequent analysis of data in real time is not possible. One reason is that data sources can fail and thus the information flow is interrupted. Furthermore there is a large divergence of the data amount generated by data streams coming from different data sources, e.g. sensors, relational databases and mobile devices. We offer an architecture to eliminate these problems. Therefore we introduce a classification of the data sources that enables an appropriate handling of the specific data source's properties.

Keywords: data integration, embedded systems, real-time, sensor networks

1 Introduction

The increasing usage of embedded systems leads to more complex heterogeneous systems. In ViERforES[1] project's context we are working with the scenario logistic hub. The logistic hub's daily business comprehends the flow of goods. Thereby, goods pass scanners, conveyors and robots, that are collecting data and controlling their flow and transfer information into central data storage systems (further details see section 2). The information is monitored and evaluated through control stations. To get the state of such complex system, e.g., the transport status or the determination of anomalies of dangerous good's actual position. There has to be found an appropriate way to run analysis on the collected information. Data Warehouses [Kim96, Inm05, BG04] provide concepts and solutions, which allow an integration as well as a flexible and efficient analysis of data. Thereby queries are usually processed on aggregated data and prepared to fit the respective department needs. Data collection and transformation processes are time consuming and additionally the data aggregation usually has to perform huge amounts of data. In some application areas, this is not possible. The problem is that, in the scenario logistic hub, the data comes from heterogeneous and distributed sources, e.g., embedded systems, and has to be gathered, aggregated and analyzed in real time. For example the logistic hub's control station is faced with problems of availability and reliability of required data. Decisions have to be made to assure that goods are delivered in-time and in-place. To support those decisions, a guarantee for real time delivery including the availability of data and processing time of analyses has to be given.

There are already Data Warehouse approaches, that provide near real time capabilities (see section 3). However, these approaches don't regard heterogeneous, distributed, and unstable data sources. A special challenge is the varying amount of input data. For example if a catastrophe on an airport takes place. That's why at this airport no airplanes can land anymore. Thus the still flying airplanes have to land somewhere else. The other airport has to handle

[1] ViERforES project web page `http://vierfores.de`

those unexpected airplanes and their data as well. Another challenge is the unreliability of the data gathering systems. Unreliability of those systems is reasoned for example by limited energy supply, e.g., wireless sensors, or they are error-prone due to environmental reasons. This situation demands special algorithms to be implemented in the extract, transform, and load (ETL) process [BG04, Inm05] to warrant a stable data quality in order to derive trustworthy information. In this paper, we propose an architecture including a classification to reduce the data amount and to compensate failures.

2 Application Scenario Logistic Hub

Within the scope of the project ViERforES, we examine different domains including the logistic hub of an airport. Therefore, in this section we give answers to the questions, which kind of systems can provide information and what are the challenges in that environment. The main task on the airfield of logistic hubs is to load and unload the airplane's cargo. Therefore, different people are involved to manage this process, like ramp agents or chief supervisors in the control station. These people need information to decide whether the airplain is ready to take off or ready to be loaded. They also decide if it has to be checked in more detail due to possible airplane's system errors. This process is assisted by embedded systems. Those embedded systems are integrated in each of the process's supporting equipment, e.g., the unit load device (ULD) that sorts the cargo to make it more manageable. In future, cargo containers are supplied with Radio-frequency identification (RFID) chips that can transmit ID values and further data. Furthermore dollies can be equipped with sensors to signify their loading status. This means, if there is anything loaded and what is loaded. Again, this information is provided by the ULDs. To enable the tracking of airplanes and dollies, GPS sensors or RFID technology can be used. Therefore, they can be located on the ramp at any time. Airplanes can automatically deliver information about their loading status, how many ULD they have loaded and, during the unloading, how many of them have already been unloaded. The shelter where dollies and other unused equipment is stored delivers information about how many empty dollies are parked. The Warehouse provides data about which ULDs have arrived and how long the unloading of the ULD had taken. Another source are reports that are supported by PDAs and filled in by ramp agents. Those PDAs can be a source as well as a sink of information. The ramp agent gets help by proposals of the unloading status generated through the system, that basically got its information out of integrated sensors.

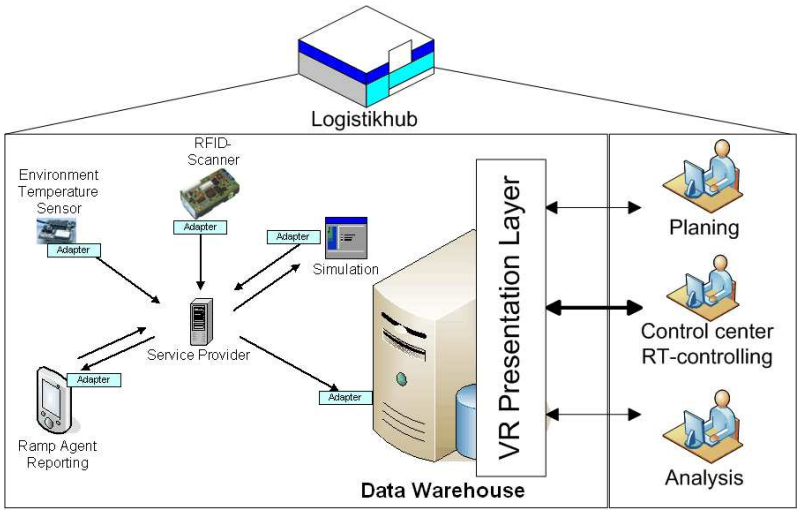

Figure 1: Sensor Network containing sensors of an airfields logistic hub

In the left part of figure 1 there are different embedded systems like those that have been described above. They are connected through a service-oriented architecture. Thereto, they have to register at a service provider, that administrates all services. The Data Warehouse is one of the registered services. In the figure's right part there are three different departments that use the Data Warehouse. One of them is the above mentioned control center.

All the before described data can be used to support the decisions made by ramp agents and the ramp control. Thereby, there are some challenges to retrieve the data. The first one is the real time challenge. On the one hand the data will be generated in time series, like position tracking, or the airplane's loading status. On the other hand data needs to be presented in real time. The ramp control needs to know the actual loading status to decide if it is ready to take off or not. Furthermore, if we consider the rising amount of data, a supporting system needs to handle a constant amount of data not only in common situations. Especially in critical situations where decisions are needed the system needs to work.

3 Real-Time Data Warehouse Architecture

This section will show problems and give an overview about existing ideas of connecting real time data with a Data Warehouse.

To be able to run analyses we assume that the data can be stored in an persistent way. Actually, this might be quite a challenge. Regarding that in our scenario the data is generated by sensors we have to take a look at what the actual amount of data might be. In [HB09] they benchmarked existing database systems. The benchmark was aimed at determining the read and write performance of existing database systems. The interesting part was and still is that just one of the tested database systems provided a sufficient performance. The others, e.g., Oracle, couldn't attain the needed performance. They assumed a data volume of 2500kb per second. If we imagine the raising usage and the potential amount of sensors on a logistic hub this data rate is easily achieved. Hence a wireless communication network is potentially used, we can assume higher data rates, e.g., the actual wireless LAN standard IEEE 802.11n.

One approach is called *trickle and flip* [Lan04]. Thereby, a copy of the original fact table is made. It is directly fed with the real time data. After a defined period of time the import of new data is stopped and this copy is copied again. The new copy is renamed to the original fact table and the old fact table is deleted. The process is called the flip. While flipping there should be no queries executed. The problem of this approach is that it is not scalable. Hence, we assume a huge amount of data this approach doesn't fit.

The idea of *table partitioning* [BG04, Rad03] is to use the feature of relational databases to create table partitions. Therefore a criteria is defined that decides about when a new partition will be created. With respect to the real time aspect this criteria will be the time. Thus the real time data will be stored in the latest partition. To prevent running queries on it, this has to be hided through the Data Warehouse. Otherwise the loading process would be disturbed. The problem in here is to find the right period of time. The aim should be that the real time partition fits in the memory. Due to the incoming sensor information the interval a day this interval can be short. Like the approach before, this approach will have a problem because the high amount of data. Thus an shorter interval must be chosen. This causes that there will be to many partitions that will decrease the performance.

Real Time partitions are introduced by Ralph Kimball [Kim02,KRT+08] to cope with the new requirement of fresh data. This approach is based on the idea to separate real time fact tables from the usual static Data Warehouse. Until the static Data Warehouse holds the historical data, the real time partition keeps the data, that was generated or changed starting at the time of the last warehouse update. Their are slightly differences in how the data is stored in the static and real time partition. It depends on what fact tables grain was chosen. The design goal should be that the real time fits into the memory. Thus the query and loading performance

can satisfy the refreshment needs demanded by the user. Although this approach is scalable the database management system hast to provide the functionality. As we have seen in the above example usual database systems have problems with the amount of data. Furthermore, the approach doesn't mention about how to load the huge amount of data and to handle unstable data sources.

The *real time data cache* [Lan04] is a completely separated database server that handles the real time data. The tables in the data cache database are modeled like the Data Warehouse tables. Additionally, just those tables that hold real time data will be created in the data cache. If analysis on the data cache's data need historical data, then this data is just in time queried from a static Data Warehouse. This approach basically has the same deficiencies like that one before.

Furthermore there is another critical aspect that needs to be considered. As usual, in real time scenarios the continuous data stream demands a way to continuously load the data. Additionally, the network load itself is another aspect. In [VS08] a workflow architecture of the ETL process is proposed. It offers a possibility to extract and load data that doesn't influence the source's and target's performance. But it doesn't regard the network load itself and it doesn't handle the information loss caused by crashed sources.

4 The Reliable Information Architecture

We have seen so far that current approaches cannot suitable handle requirements arising in complex systems that have to face reliability problems. Therefore, we propose in our architecture (see figure 2) a classification of the data providing sensors or embedded devices respectively. We define three classes:

- *Preprocessor class* contains data sources, e.g., embedded devices, database services, and sensors that locally preprocess the data, i.e., directly on the device. This is needed due to a too large amount of data that has to be transferred. Preprocessing methods include aggregation functionality, data cleaning, etc.
- *Intermediate Aggregation class* contains nodes, where data is aggregated. Furthermore, nodes that send data to be aggregated belong to this class as well. Using these aggregation nodes we can significantly reduce the network load. Additionally, an increased reliability of data is achieved due to a compensation of single node failures.
- *Direct Input class* are those data sources transferring their data directly to the Data Warehouse. Examples are time critical systems and traditional Data Warehouse sources. Thus, mission critical data can be provided straight forward to the control center.

As an example let us take embedded systems that are measuring temperature. We assume that basically the temperature sensors of the system provide data 5 times a second. Now, devices have different possibilities to handle the data. A device would be classified as preprocessor if it aggregates the temperature over a period of 5 minutes locally. This value is send to the Data Warehouse. In another case there are some devices that don't send their measured values to the Data Warehouse either preprocessed or directly. They transfer them to an intermediate node which gets this data and values from other sub devices too, e.g., a hall's temperature determination node that aggregates the values from different measures. Those measures take the temperature of heat sources in that hall. Both the node that sends data to the intermediate node and the intermediate node itself are members of the intermediate aggregation class. Another device, e.g., used in more critical area with for example temperature sensitive liquids transfers the data directly to the Data Warehouse as it comes. Those device are members of the direct input class.

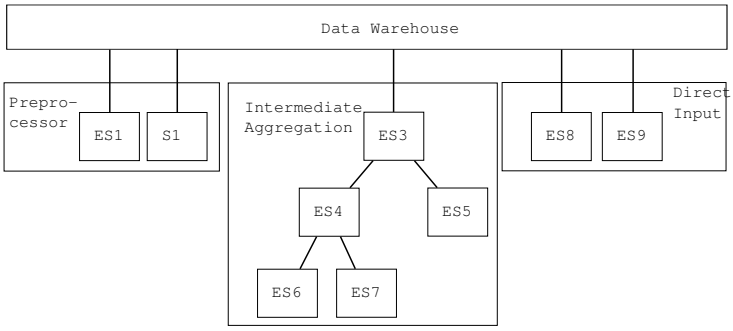

Figure 2: The Reliable Integration Architecture

This classification has two major advantages: (1) the reduced amount of incoming data and (2) the reduced network load. Additionally, it can be used to evaluate existing architectures. This information can help to make changes and extensions. Furthermore, based on the classification, we propose different storage and processing functions to overcome the mentioned problems.

5 Conclusion

In this paper we have shown problems of the real time decision support in today's complex systems. Therefore, we have taken the scenario logistic hub as an example. Afterwards, we have discussed existing approaches to store and transfer real time data and have shown their deficiencies regarding the mentioned problems. Hereupon, we have proposed a reliable information architecture that offers a classification of existing systems and eliminates the mentioned problems. The future work encloses an implementation and evaluation of our proposed architecture.

References

[BG04] Andreas Bauer and Holger Günzel. *Data Warehouse Systeme: Architektur, Entwicklung, Anwendung*. Heidelberg:dpunkt-Verlag, 2te, überarbeitete und aktualisierte auflage edition, 2004. ISBN 3-89864-251-8.

[HB09] Daniel Herold and Wolfgang Benn. Sensordatenablage in echtzeit. *Datenbank Spektrum*, 28:31–36, Feb 2009.

[Inm05] William H. Inmon. *Building the Data Warehouse*. New York:John Wiley & Sons, 4th edition, Oct 2005. ISBN 978-0-7645-9944-6.

[Kim96] Ralph Kimball. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. Wiley & Sons, 1996. ISBN 0-471-15337-0.

[Kim02] Ralph Kimball. Realtime partitions, February 2002. Intelligent Enterprise Magazine, `http://www.intelligententerprise.com/020201/503warehouse1_1.jhtml`.

[KRT+08] Ralph Kimball, Margy Ross, Warren Thornthwaite, Joy Mundy, and Bob Becker. *The Data Warehouse Lifecycle Toolkit*. Wiley & Sons, 2nd edition, January 2008. ISBN 978-0470149775.

[Lan04] Justin Langseth. Real-time data warehousing: Challenges and solutions, 2004. DSSResources.COM, `http://dssresources.com/papers/features/langseth/langseth02082004.html`.

[Rad03] Neil Raden. Real time: Get real, part ii, June 2003. Intelligent Enterprise Magazine, `http://www.intelligententerprise.com/030630/611warehouse1_1.jhtml`.

[VS08] Phano Vassiliadis and Alkis Simitsis. *Near Real Time ETL*, volume 3 of *Annals of Information Systems*, chapter 2. Springer US, 2008. ISBN 978-0-387-87430-2.