Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium, Volume 22, No. 1, ISSN 1726-9679
ISBN 978-3-901509-83-4, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2011
Make Harmony between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals & Proceedings of DAAAM International 2011

DB4OBJECTS BASED BUFFERING APPLICATION FOR USE IN SOFTWARE MONITORING SYSTEMS

GRIGORESCU, C[ostin] - M[arius]; MORARU, S[orin] - A[urel]; KRISTALY, D[ominic] M[ircea] & BADEA, M[ilian]

Abstract: *Buffering inside a distributed monitoring software system that runs inside a LAN and stores readings on a central component is a very important issue. This paper presents the structure of a buffering application that is based on a DB4Objects object database. The application is designed to operate inside a distributed monitoring software system between the data acquisition and server components. Its main purpose is to offer persistent storage for the readings in the case of communication problems between the two components. When the communication problems are fixed, the application resumes sending real time readings and starts transferring the buffer's contents to the server. For optimal performance, the buffer is divided in partitions that contain the readings associated to one hour.*

Key words: *buffering, object databases, monitoring system*

1. INTRODUCTION

In general, distributed monitoring software systems have a data acquisition component, a data storage component and a data display component (Wikipedia, 2010).

This paper focuses on the communication between the data acquisition and data storage components. The components of a distributed monitoring software system run on different computers and send data between them using, in most cases, the Ethernet interface.

The data flow between the data acquisition component and data storage component consists of readings made by the first one that need to be stored inside databases or other storage systems by the second one. If this flow gets interrupted, then the readings are lost and the monitored information will contain inconsistences. The flow can be interrupted by various reasons like software and hardware problems, electrical problems, LAN problems, etc.

In the case of software, hardware or electrical problems on the server side and LAN problems, a buffering application running on the same computer as the data acquisition component represents a solution for temporarily storing the readings while the server side or LAN problems are fixed (Grigorescu et al., 2010).

This paper presents, in its following sections, a buffering application concept based on a DB4Objects object database. The following sections present the concept of object databases with advantages and disadvantages and a speed test performed on DB4Objects and the concept of the buffering application with work-flow diagrams for the main actions performed by it.

2. OBJECT DATABASES

An object database is represented by a database management system in which the data is stored using objects similar to the ones used in object oriented programming.

The decision to use an object oriented database system was influenced by the following main aspects: **integrability** (the need to be able to integrate the buffering application with almost no changes to the existing monitoring system), **configurability** and **easy installation** (the need to develop an application that needs no configuration or specific installation for the buffer storage) and **easy maintenance** (the need to be able to perform maintenance tasks easily) (Wikipedia, 2011).

Choosing the DB4Objects database solution resolved most of the above aspects such as:

- Configurability and instalation: DB4Objects does not require an installation procedure. It comes as a JAR archive for Java or a DLL library for .Net. After referencing one of these libraries inside a project, the users can benefit from all the features provided by DB4Objects.
- Maintenance: The DB4Objects databases are stored in single files which makes any backup procedure very easy to perform.

After selecting the object database type, the next step was to test its performance by running queries on different database configurations. These configurations used different partitioning techniques such as daily partitions and hourly partitions.

Daily partitioning consists of one database for each day and hourly partitioning consists of one database for each hour of a day. For example, for the month of March a daily partitioning system will contain 31 databases and an hourly partitioning system will contain 744 databases (31 days * 24 hours per day).

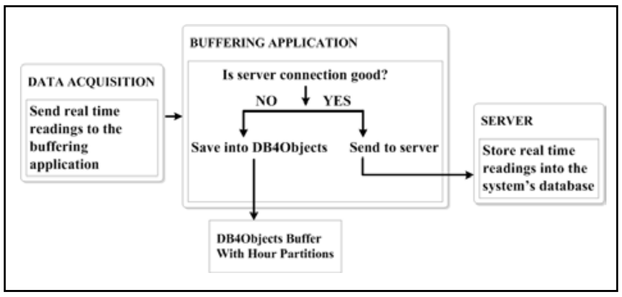
An application was developed to generate readings that were stored in DB4Objects databases with daily and hourly partitions. This application generated readings for 8 months (January - August) with a 1 second interval between each reading (3600 readings per hour, 86400 readings per day). For daily partitioning the application generated a total of 243 databases and for hourly partitioning it generated a total of 5832 databases (Versant Corporation, 2011).

The performance test consists of running select queries for different periods of time and measuring the time needed to get the results. These periods of time are: 15 minutes, 30 minutes, 1 hour and 1 day.

The average time needed by DB4Objects to return the objects is presented in Tab. 1. This time shows that, for a small number of readings, the hourly partitioning system is 4 times faster than the daily one. Considering this result, the hourly partitioning system was selected.

Partition Type	*T1 (ms)*	*T2 (ms)*	*T3 (ms)*	*Avg (ms)*
Time to get the readings for 15 minutes				
Daily	3578	3328	3547	3484.3
Hourly	891	984	844	906.3
Time to get readings for 30 minutes				
Daily	3469	3390	4047	3635.3
Hourly	859	891	922	890.6
Time to get readings for 1 hour				
Daily	3500	4250	3625	3791.6
Hourly	875	859	1031	921.6
Time to get readings for 1 day				
Daily	3328	4141	3812	3760.3
Hourly	6172	5547	5125	5614.6

Tab.1. Performance tests results


BUFFERING APPLICATION
DATA ACQUISITION
Send real time readings to the buffering application
Is server connection good?
NO
YES
Save into DB4Objects
Send to server
SERVER
Store real time readings into the system's database
DB4Objects Buffer With Hour Partitions
Fig. 1. Buffering application workflow

3. BUFFERING APPLICATION CONCEPT

Fig. 1 presents the workflow of the proposed concept: the application receives readings from the data acquisition component and sends them either to the data storage component or to the DB4Objects buffering system.

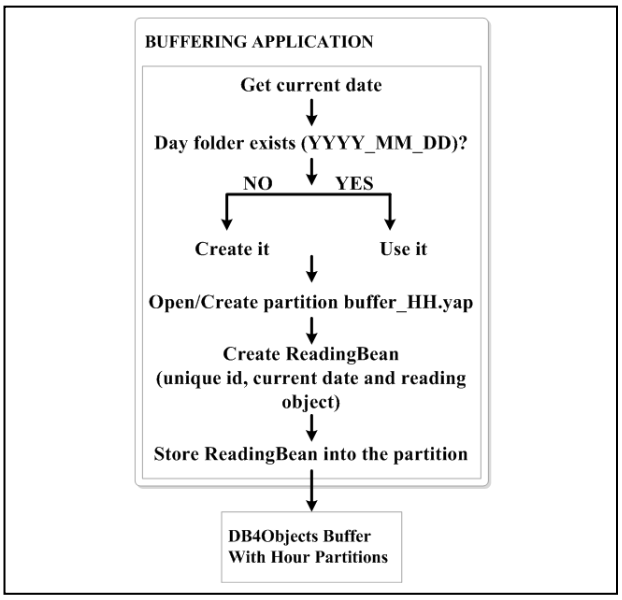
Because the buffering application is located between the data acquisition and storage components it needs to communicate with both of them. This communication is made using sockets: an input socket which is listened for incoming data and an output socket (on the server side) which is periodically checked by a thread (Mahmoud, 1996). If the output connection is alive, then all readings received from the data acquisition component are sent directly to the data storage component.

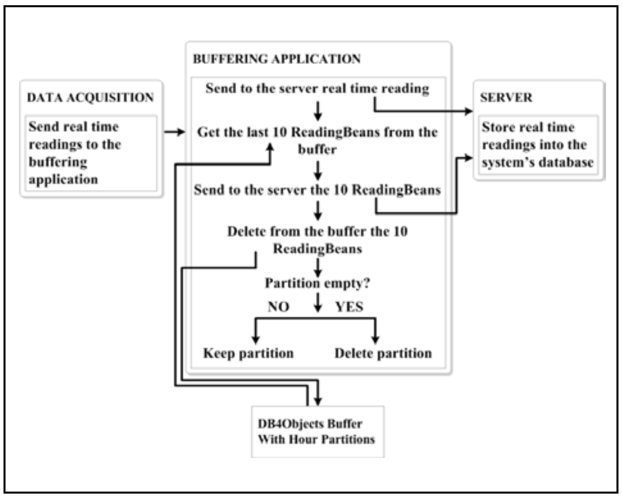
If the connection is not alive, then the application enters the buffering operation mode presented in Fig. 2.

In this operating mode, the application follows the following steps:

- Gets the current date in order to check if a folder named *YYYY_MM_DD* already exists. In this folder all hour partitions for the specified date are kept. If the folder does not exist, then the application creates it.
- Opens a connection to a DB4Objects database named *buffer_HH.yap*. If the database does not exist, then DB4Objects will automatically create it.
- Creates a *ReadingBean* object in which it will add the following information: current date and time, reading object received from the data acquisition component and a unique identifier.
- Stores the *ReadingBean* object inside the DB4Objects database and closes the connection with it.

The reading object can have any structure as long as it can be a child of the *Object* class from Java.


BUFFERING APPLICATION
Get current date
Day folder exists (YYYY_MM_DD)?
NO
YES
Create it
Use it
Open/Create partition buffer_HH.yap
Create ReadingBean (unique id, current date and reading object)
Store ReadingBean into the partition
DB4Objects Buffer With Hour Partitions
Fig. 2. Buffering operating mode


BUFFERING APPLICATION
DATA ACQUISITION
Send real time readings to the buffering application
Send to the server real time reading
Get the last 10 ReadingBeans from the buffer
Send to the server the 10 ReadingBeans
Delete from the buffer the 10 ReadingBeans
Partition empty?
NO
YES
Keep partition
Delete partition
SERVER
Store real time readings into the system's database
DB4Objects Buffer With Hour Partitions
Fig. 3. Post buffering operation mode

This solves the integrability aspect mentioned before because the buffering application does not need to know the structure of the reading objects that are sent from the data acquisition component to the data storage one.

Fig. 3 presents the post buffering operation mode. The application enters in this mode after detecting that the connection with the data storage component has been reestablished.

The steps of this operating mode are:

- Send a real time reading object to the server.
- Get the last 10 *ReadingBean* objects from the buffer and send their reading objects to the server. If the objects were successfully sent, then delete the *ReadingBeans* from the buffer.
- If the partition is empty, then delete the partition.

4. CONCLUSION

This paper tries to propose a concept for a buffering application based on an object database system provided by Versant: DB4Objects. The role inside a monitoring system of this buffering application is to prevent monitored data loss in the case of a communication failure between the component that makes the data acquisition and the component that stores and manages it.

5. ACKNOWLEDGEMENTS

This paper is supported by the Sectorial Operational Programme Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the contract number POSDRU/88/1.5/S/ 59321.

6. REFERENCES

Grigorescu, C. M.; Moraru, S. A.; Neukart, F. & Badea, M. (2010). Buffering application for an industrial monitoring software system, *Proceedings of the 12th International Conference on Optimization of Electrical and Electronic Equipment, OPTIM 2010*, 20-22 Maz 2010, Brasov, Romania, 971-1-4244-7020-4/10, pp. 780-780

Mahmoud, Q. H. (1996). Sockets programming in Java: A tutorial, JavaWorld.com

DB4Objects 8.0 Java Reference Guide (2011). Versant Corporation

*** (2011) http://en.wikipedia.org/wiki/Object_database - Wikipedia, *Accessed on: 2011-07-20*

*** (2010) http://en.wikipedia.org/wiki/Distributed_computing - Wikipedia, *Accessed on: 2011-07-18*