ADAPTIVE SYNCHRONIZATION OF BUSINESS OBJECTS IN SERVICE ORIENTED ARCHITECTURES

Michael Ameling, Bernhard Wolf
SAP Research CEC Dresden, Chemnitzer Str. 48, Dresden, Germany

Thomas Springer, Alexander Schill
Dresden University of Technology, Noethnitzer Str. 46, Dresden, Germany

Keywords: Application server, Business objects, Replication, Synchronization, Web Service, SOA.

Abstract: Business applications such as supply chain management and enterprise relationship management use business objects (BOs) for data containers. The BOs are cached at the middle-tier since the applications are hosted on application servers within a multi-tier architecture. The applications are replicated to achieve scalability and fast local access for the clients. Therefore, replica control for the BOs is mandatory to fulfill consistency requirements. However, following the service-oriented architecture the synchronization of BOs through standardized services is time consuming and can be optimized. In this paper, a solution is presented that allows an adaptive synchronization for business objects based on profiling. A BO and system profiling enables an efficient synchronization by an appropriate configuration of the replication strategy. A cost model based on an experimental evaluation allows to find e.g., the trade-off of sending full BO copies or just delta synchronization messages. The proposed solution is evaluated by temporal consistency constraints for BOs. Finally, an initial configuration of the replication strategy and an adaption during runtime is applicable based on constantly updated profiles.

1 INTRODUCTION

Recent software solutions for mid-size companies provide functionalities of typical business applications e.g., Customer Relationship Management (CRM), Project Management (PM) and Supply Chain Management (SCM). Applications are hosted typically on application servers within a multi-tier architecture which allows resource sharing and thin client support. However, the number of clients can be very high. High activity of users can lead to low performance of the applications. Furthermore, clients can be globally distributed which results in long latencies.

Replication is a solution to achieve scalability and provide fast local access. Several instances of business applications can be hosted at one application server or might even be replicated across different application servers. Since the replicated data within the applications has to be up-to-date and consistent replica control is strongly required.

Common solutions provide replication at the database level where replica control has been investigated very well over the last two decades. However, data containers in business applications are large and complex business objects (BOs). The BOs provide services to be read, created, modified and deleted. They are involved in business processes and link to other BOs. Especially, data belonging to one BO is usually distributed over multiple database tables. Read and write access is provided by services and always executed in the context of business operations. A replication at database level has to operate on its schema specifications only, without any assumption about the application context the data is changed in. This results in a loss of application knowledge and the separate handling of information that belongs together. Furthermore, the application servers do not necessary access the same database type at the persistence layer. Therefore, the replication of BOs at application level is aimed where the application context can be considered and existing services can be used. The synchronization of BOs is time consuming since BOs are complex data containers to be processed and their size leads to a large data volume to be transferred. A synchronization always takes place from the changed BO at the *primary* (called *sender*) to the replicated BO at the *secondary* (called *receiver*). The replication strategy has to be carefully selected. In example the trade-off either to send the full copy or the delta of a changed BO has to be found. The two cases are listed in Table 1. In Case I, copying the BO at the sender and a replacement at receiver side is not very time consuming. On the other hand, a message including a full copy results in long transfer time. In Case II, the sending of a delta means additional effort at sender and receiver for processing the delta of the BO and integrating the changes. However, this way the message size can be reduced significantly, if just a subset of the BO data has been changed. Therefore, transfer time can be saved compared to case I.

Table 1: Trade-Off Full Copy vs. Delta.

Case	Sender	Network	Receiver
I.	copy full BO	transfer copy	replace BO
II.	process delta	transfer delta	integrate delta

Our solution focuses on an adaptive synchronization for BOs at the middle-tier to support an efficient update process for replicated BOs. We consider a primary copy approach where write access is only granted to one master BO. We introduce a *profiling of BOs* and a *profiling of the system environment* to achieve an efficient synchronization process exploiting knowledge about the BO structure and application level context of change. In order to achieve efficiency we provide a *cost model* based on a *BO model* used for profiling of BOs and a *system model* used for profiling of the system environment. In Fig. 1 the conceptual steps of our approach are depicted. The first step (1) is the *profiling of BOs* based on the BO model. BO instances are profiled individually. The second step (2) is the *determination of system parameters*. The profile of each system is stored persistently as well. The third step (3) is the *execution of the cost model* where the BO profiles and system profiles are used to determine the costs for the synchronization processes based on our cost model. In step four (4) the result of step three can be used to choose and configure the replication strategy to achieve an efficient replication process. Since BOs are changing, BO and system profiles have to be updated and replication parameters have to be adjusted accordingly. Step (5) performs an *adaptation of replication parameters* during runtime.

The rest of the paper is organized as follows: In Section 2 the synchronization of BOs in SOA is described in detail. It is followed by related work in Section 3. Afterwards, we follow the conceptual steps

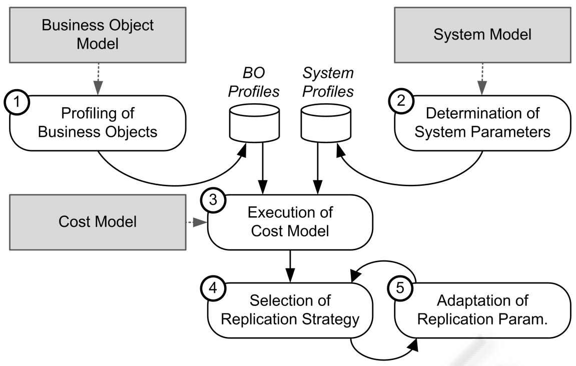
Business Object Model
System Model
BO Profiles
System Profiles
1
Profiling of Business Objects
2
Determination of System Parameters
Cost Model
3
Execution of Cost Model
4
Selection of Replication Strategy
5
Adaptation of Replication Param.

Figure 1: System Support for Efficient Synchronization of Business Objects.

in separate sections as depicted in Fig. 1: profiling of BOs in Section 4, determination of system parameters in Section 5, execution of cost model in Section 6 and the selection of the replication strategy in Section 7. In Section 8 a short conclusion and outlook is given.

2 SYNCHRONIZATION OF BUSINESS OBJECTS IN SOA

According to a multi-tier architecture the application logic is executed at the middle-tier. At this tier, BOs are instantiated and cached for local access and efficient processing. This set-up allows implementing agents watching BOs for changes. A change pointer which references changed BOs is advisable since agents slow down the response time for clients. However, synchronization agents at sender and receiver include the replica control observing the BOs for changes, assembling synchronization messages and integrating the changes. Following a service-oriented architecture (SOA) the business logic of the applications is exposed through well defined interfaces. The same applies for replication processes such as the synchronization of BOs. At application level Web Services (W3C, 2002) are used to send synchronization messages via the network.

In widely distributed systems the lazy primary copy approach is often used (Ameling et al., 2008). Primary copy performs well for read intensive applications and simplifies concurrency control (Gray et al., 1996). Modifications can be performed on BOs at the primary (master) only. Secondaries have to pass changes to the primary. The goal is to keep all BOs consistent. We assume that all BOs at secondaries have the same state. Therefore, the same synchronization message is sent to all secondaries (receivers).

Summarized, the synchronization of BOs in SOA can be distinguished in processes taking place at the

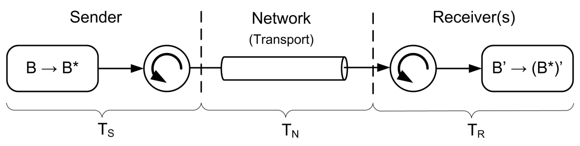

Figure 2: Synchronization Process.

sender, the transport of the synchronization message and the processes taking place at the receiver. A confirmation message is not mandatory within an optimistic approach and therefore not considered. In Fig. 2 a time based division in sender (T_S), network (T_N) and receiver (T_R) is depicted. T_S concludes the time for all processes at the sender. It starts once the transaction of a client is committed and the BO B was changed to BO B^*. It ends when the synchronization message is assembled and passed to the infrastructure. T_N concludes the time for the transport of the synchronization message. Several receivers can be addressed in parallel. Therefore, the longest transport time is relevant to reach consistency of the complete system. T_R concludes the time for processing the message and updating the BO B' to $(B^*)'$ at one receiver. The synchronization process is finished when all replicas have the same state as the primary.

3 RELATED WORK

The synchronization of BOs requires an understanding of replica control at application level. We took advantage of the replication strategies well known for databases ((Pacitti et al., 1999), (Pedone et al., 2003), (Plattner et al., 2007)). Approaches implementing a replication at the middle-tier are Middel-R (Marta Pati et al., 2005), Ganymed (Plattner and Alonso, 2004) and CORBA ((Othman et al., 2001), (Killijian and Fabre, 2000)) but mainly focus on one single algorithm or the replicated application servers share one single database. The introduced primary copy approach is used in research solutions (Felber and Narasimhan, 2002), (Barga et al., 2002) as well as industrial solutions such as JBoss and WebSphere since it performs well for read intensive applications. Most approaches are primarily designed to achieve fault-tolerance ((Wu and Kemme, 2005)). The only approaches mainly focusing on the replication of BOs are (Salas et al., 2006) and (Perez-Sorrosal et al., 2007) but use specific algorithms. In (Ameling et al., 2009) a cost model for an efficient BO replication was introduced which is included in the proposed solution. Furthermore, a framework for simulating the configuration of different replication strategies was introduced in (Ameling et al., 2009).

4 PROFILING BUSINESS OBJECTS

The schema of a BO defines which elements belong to a BO, where they are placed and which elements are mandatory or optional. The choice of optional elements and cardinalities do not allow determining BO structure. In the following a BO model is introduced that allows a profiling of the structure and further parameters of BO instances.

4.1 Business Object Model

The BO model defines the parameters used for profiling BOs. It covers the parameters that influence the synchronization process. In combination with the cost model a recommendation for the configuration of replication strategies can be provided. In (Ameling et al., 2009) we introduced a cost model for an efficient BO replication based on the structure of BOs. Therefore, we defined a *structure model* for BOs. The following BO model includes the structure model as one parameter set. Further parameters are completeness, access ratio and occurrences of BOs. An introduction of all BO model parameters follows.

The *structure* of BOs allows to determine processing times for BOs at sender and receiver side. A determination on processing times based on the size of BOs only is not possible since the structure has significant influence on the overall synchronization time (Ameling et al., 2009). The structure model includes the number of elements, their size, and their position within the BO. Therefore, T_S and T_R based on the data of synchronization messages can be provided.

The *completeness* expresses the use of mandatory elements according to the schema. Therefore, empty values and non used elements can be identified. Since optional elements and cardinalities increase the use of elements an additional value for completeness including optional elements and cardinalities has to be profiled as well.

The *access* to BOs by client applications plays a significant role for the synchronization process. Read access affects priority of synchronization of BOs. A small *read ratio* might be prioritized as low. Synchronization of often read BOs can be defined with higher priority. On the other hand, the *write ratio* has influence on the amount of synchronization messages. Each write implicates a change which results in a need for synchronization.

The value *occurrences* indicates the number of BO instances existing within a system. It is the only value not related to one BO instance but to BO classes. It has influence on e.g., the number of messages to expect.

Finally, all the introduced parameters compose the BO model which is the foundation for BO profiling. Each profile of a BO holds all parameters of the BO model. An example profile is given in Table 2.

4.2 Profiling of Business Object Instances

BO instances are profiled to get the structure model parameters. The other parameters of the BO model are only determinable from BO instances as well.

In Table 2 an example of a *Sales Order* profile is given. A selection of structure model parameters (Ameling et al., 2009) are listed: N - number of nodes, L - number of levels, K - number of attributes, Λ - maximum number of positions, W - size of all attributes in bytes, V - total size of leaf nodes, F - number of links, and a selection of N_l - number of nodes at level l. Furthermore, values for the size of the complete BO, the completeness (incl. optional elements and cardinalities in parentheses), read ratio, write ratio and occurrences are listed.

Table 2: Example Sales Order Profile.

STRUCTURE							
N	L	K	Λ	W	V	F	...
865	27	1,345	27	899,455	13	12	...
N_1	N_2	N_3	N_4	N_5	N_6	N_7	...
3	12	43	23	44	33	12	23
SIZE	380.233 *Byte*						

COMPLETENESS	63% (391%)

ACCESS			
read ratio	3.78 h^{-1}	write ratio	1.18 h^{-1}

OCCURRENCES	589,333

The profiling of the BO's structure can be done for all BOs right after the instantiation of the BO. During runtime profiles have to be updated when BOs were changed. The time for updating profiles can be ignored for write operations. The Algorithm 1 describes how to create a structure profile for a node. Algorithm can be used iteratively to get a complete structure profile of a node. The parameters for number of nodes (N), levels (L), attributes (K) and links (F) are collected. The size of attributes (W) and values (V) are determined as well. Furthermore, the parameters for certain nodes, positions and levels have to be collected. Therefore, the indexes l (level), n (node) and λ (position in a node) are used.

Algorithm 1: *getProfile(nodes, level)* determination of structure profile parameters for a node.

```
Require: node $n$, level $l$
 1: $N++$;
 2: $N_l++$;
 3: if node contains attributes then
 4:     $\lambda = 0$;
 5:     for all attribute $k$ do
 6:         $K++; K_l++; K_n++; K_\lambda++; \lambda++$;
 7:         $W = W + W(k); W_l = W_l + W(k)$;
 8:         $W_n = W_n + W(k); W_{n,\lambda} = W_{n,\lambda} + W(k)$;
 9:     end for
10: end if
11: if node contains sub nodes then
12:     if $L < l + 1$ then
13:         $L++$;
14:     end if
15:     for all sub node $u$ do
16:         $getProfile(u, l + 1)$;
17:     end for
18: else if node has value then
19:     $V = V + V(v); V_l = V_l + V(v); V_n = V_n + V(v)$;
20:     if value is a link then
21:         $F++; F_l++$;
22:     end if
23: end if
```

The update of the structure profile works similarly. Adding and deleting of elements increases and decreases parameter values. Modification of elements only affects the parameters for size of elements. The read ratio is logged during system operation. The write ratio can be determined from timestamps and versioning of BOs. We assume that the schema of a BO is available. Therefore, the completeness can be determined with a comparison of the structure profile and the BO schema.

4.3 Business Object Analysis Tool

The profiling of BOs can be done within the application or as a loosely coupled service. We developed a Business Object Analysis Tool (BOAT) that provides profiling as a Web Services. BOs can be sent as XML documents in the request. The response is a BO profile.

BOAT allows to group profiles. Therefore, it is possible to make general assumptions about BOs. In example profiles can be grouped by a BO type or even for a specific domain. A user interface providing chart views is implemented. The schemas of BOs are stored in a repository to allow a comparison and to determine the completeness.

In Fig. 3 the scenario for profiling BOs from an application including the architecture of BOAT is depicted. The applications allow requesting BOs through services (a) (1). The services (b) provided

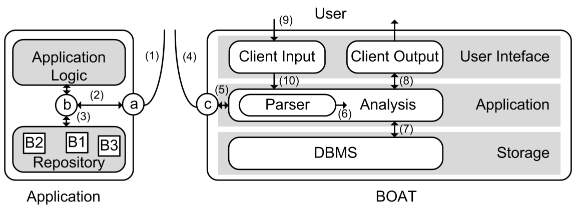

Figure 3: Business Object Analysis Tool.

by the BOs (*B1, B2, B3*) themself are used to get full copies (2) and finally return the BO instances. The BOs can be stored locally at any application. BOAT provides the Web Service (c) to profile BOs (4). Once the Web Service is called the BO document is processed (5) and an analysis is done (6). The result in form of a profile is stored persistently (7). The response of the Web Service (c) includes the profile. BOAT provides a user interfaces which allows e.g., a statistical analysis of profiles as mentioned above.

5 DETERMINATION OF SYSTEM PARAMETERS

5.1 System Model

The system model provides the parameters describing the costs for a single process for synchronization. In Fig. 2 we already divided a BOs synchronization into processes at sender, network and receivers. For simplification we assume that each receiver behaves equally.

At the sender the parameters for identifying changes, parse the BO and message assembling are crucial. The first parameter describes the time an agent needs to compare a new version of a BO with the previous version of that BO until all changes are identified. The parsing of the BO can be determined by single parameters for processing each element of a BO. The parameters for the processing time for a node a, of an attribute b, of a Byte of an attribute value c, of a Byte of an node value d and for resolving a link e are used. Further parameters are for identifying changes and message assembling e.g., creation of a SOAP message.

For the transport process we use the bandwidth B_W and latency L_W as system model parameters as introduced in (Ameling et al., 2008). Additionally the replication factor R_F is one parameter which describes the number of receivers to be addressed.

At the receiver the parameters for the processing time of the message has to be defined. Parameters for the parsing of the received synchronization message are the processing time for a node, attribute, etc. Since changes have to be integrated the parameter for this step has to be defined as well.

A higher granularity for the processes at sender and receiver side exists but is not focus in this paper. Additional processes during synchronization require further parameters. In example a process for security check can be added which requires an additional parameter. However, since the single process steps for a synchronization are executed sequentially parameters can be added easily.

5.2 Parameter Determination

The parameter determination to provide a system profile can be done before operational mode. A profile includes the costs for processes at sender and receiver. The transport parameters are included as well.

The determination of the parsing parameters was done in an experimental evaluation. In Table 3 an example of a system profile with a selection of parameters is given. Synchronization process steps at sender and receiver side have to be measured during execution. The bandwidth and latency for transportation of synchronization messages have to be determined by sending normalized messages.

Table 3: Example System Profile.

SENDER				
a_l *in ns*	b_λ *in ns*	c *in ms*	d *in ms*	e *in ns*
43.53 l	47.596 λ	106.24	145.87	90.684
TRANSPORT				
B_W *in* $MBit/s$		L_W *in ns*	R_F	
5.443		127	12	
RECEIVER				
a_l *in ns*	b_λ *in ns*	c *in ms*	d *in ms*	e *in ns*
21.765 l	23.798 λ	58.119	72.934	45.342

In the sender profile the parameters for the processing time of a node in dependency of the level a_l, the processing time for an attribute in dependency of the position within a node b_λ and the other determined parameters are listed. We determined the parameters by measuring BO documents that were individually created. This way, we were able to solely modify the number of one type of element. Afterwards, we measured the processing time of each created BO document and were able to determine each parameter. Since different parsers are possible a comparison of different implementations parsers was done. The implementation also affects the dependency of the processing time on the structure. The listed result just

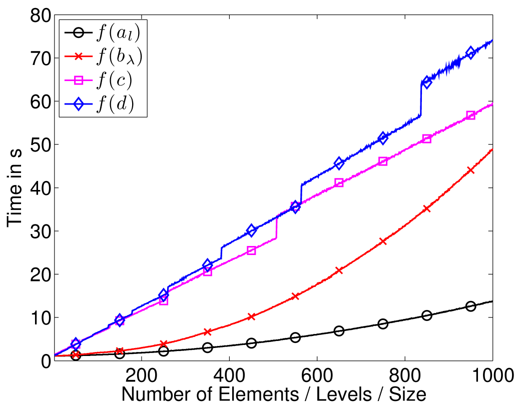

Figure 4: System Profiling.

present one set of the results but validate that we are able to determine the parameters with our solution.

The determination of the system profile can be done with an experimental setup which is detailed described in (Ameling et al., 2009). In Fig. 4 the measurements for increasing the number of nodes $f(a_l)$, the number of attributes $f(b_\lambda)$, the size of attribute values $f(c)$, and size of node $f(d)$ are depicted. The intersection with the ordinate is the value for processing a BO without content. The method of least squares allows determining functions out of the discrete measurements. In this way e.g., a function for the processing time of a node in dependency of the level $f(a_l)$ can be determined. The function $f(c)$ shows a linear increase for an linear increase of the attribute value. The slope exactly reflects the processing time of a Byte of an attribute value (parameter c). In Table 3 the results of the experiments for the parameter determination are listed.

6 EXECUTION OF COST MODEL

In the previous sections the BO and the system model were introduced. The models are used for profiling BOs and the systems used in the replicated environment. In (Ameling et al., 2009) the part of the cost model for determining the parsing time of BOs was introduced. It uses the structure parameters of the BO profile. The following equation enables to calculate the processing at the sender side:

$$T_S = T_{offset} + \sum_{l=0}^{L}(N_l \times a_l) + \sum_{\lambda=1}^{K_n}(K_\lambda \times b_\lambda) + W \times c + V \times d + F \times e$$

The defined cost model presents a sum of the time needed for all process steps for one synchronization of a changed BO. The lazy primary copy approach for replication is used. Therefore, we consider the synchronization from a master to the replicas. However, our cost model does also apply for the use of other replication strategies. In example, a termination strategy does require an additional confirmation message resulting in additional message assembling at receiver side, an additional transport of that message and a processing at the sender. A switch to the update every approach can be covered (Gray et al., 1996). Additional process steps can be easily inserted into the sequence of synchronization process steps. A comparison between different replication strategies is planned for future work. The current focus is the configuration of the currently used replication strategy.

The validation of the cost model was done experimentally with the use of dumps of BOs. BOAT was used to do a profiling of BOs. To avoid an implementation within a live system the replication environment introduced in (Ameling et al., 2009) was used. Therefore, we were able to determine processing time of BOs and to simulate a synchronization of changed BOs. The determination of read and write ratio usually requires a comprehensive observation of representative applications.

7 SELECTION OF REPLICATION STRATEGY

The cost model including profiling is used to support an efficient configuration of the synchronization process. The following two examples for the selection of the replication strategy are given: the sending of a *full copy* of a BO verses the sending of the *delta* (amount of changes between two different BO versions) and the *bulking* of BO changes. Both strategies can be combined.

7.1 Full Copy vs. Delta

To reach more efficiency the time efforts for *case I* and *case II* have to to be compared (Tab. 1) to choose the most beneficial. Therefore, the processing times T_P and the transfer times T_N for both cases have to be determined. The processing time T_P summarizes the process steps of the synchronization at the sender (T_S) and at receiver (T_R). All values for *case I* ($T_P(full), T_N(full)$) and *case II* ($T_P(delta), T_N(delta)$) can be determined with the introduced cost model based on profiling.

The decision either to use *case I* or *case II* depends on the trade-off between the time that can be saved at transfer (ΔT_N) and the additional effort for processing (ΔT_P). In Fig. 5 a method to describe the trade-off and to find the break-even-point to switch between the replication strategy is depicted. The method considers the relative size of the delta

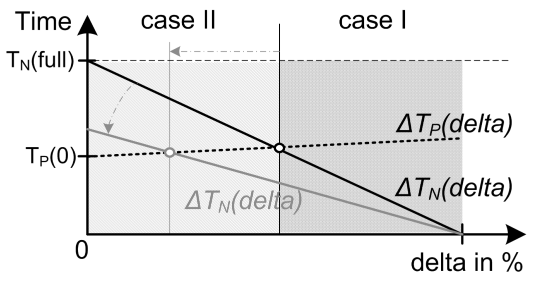

Figure 5: Decision Model.

compared to the full size of the BO. The function $\Delta T_P(delta)$ defines the processing time that can be saved sending a *full* copy (*case I*) where $\Delta T_P(delta) = T_P(delta) - T_P(full)$. The function $\Delta T_N(delta)$ defines the transfer time that can be saved sending a *delta* message (*case II*) where $\Delta T_N(delta) = T_N(full) - T_N(delta)$. For simplification we assume that the used BO has a fixed size. Therefore, both functions depend only on the size of the delta since the size of a full copy is constant if no additional elements are added to the BO ($T_N(full) = const.$).

The function $\Delta T_N(delta)$ decreases with a larger delta size since less transfer time can be saved sending a delta message. The intersection with the ordinate is the transfer time of a full copy $\Delta T_N(0) = T_N(full)$. It equals the maximum transfer time that can be saved. No time can be saved when the delta equals a full copy ($\Delta T_N(100\%) = 0$). The function $\Delta T_P(delta)$ lightly increases since a larger delta results in more effort for processing in *case II*. The intersection with the ordinate is the processing time for an empty delta message ($T_P(delta)$). Finally, the intersection of the functions $\Delta T_N(delta)$ and $\Delta T_P(delta)$ equals the delta value to switch between the two cases. If the saved transfer time $\Delta T_N(delta)$ sending a delta exceeds the saved additional effort $\Delta T_P(delta)$ sending a full copy then *case II* is more efficient (*case II*: $\Delta T_N(delta) > \Delta T_P(delta)$ - left side). Visa versa *case I* is used if $\Delta T_P(delta)$ exceeds $\Delta T_N(delta)$ (*case I*: $\Delta T_P(delta) \geq \Delta T_N(delta)$ - right side).

The previous example was given for one BO with a fixed size. The method to find the trade-off considers the relative delta. The structure of the BO was fixed and is reflected in the cost model. Let's assume another BO with a smaller size is changed. For simplification the function for the saved processing time $\Delta T_P(delta)$ does not change due to e.g., more complex structure. The smaller size of the BO results in a decrease of $T_N(full)$. The function $\Delta T_N(delta)$ is below the previous function (gray line in Fig. 5). Finally, the intersections of the functions $\Delta T_P(delta)$ and $\Delta T_N(delta)$ moves to a smaller delta resulting in an earlier switch from *case II* to *case I*. The introduced cost model and the profiling allow predicting both functions for any BO. Finally, we are able to decide either to send immediately a full copy or a delta message for each BO based on the size of the delta and the structure of the BO.

7.2 Bulking

A high write ratio of BOs causes frequent synchronization messages which can result in high network traffic. Consistency constraints allow that not synchronized BOs are still valid for e.g., a certain amount of time. Therefore, several changes can be bundled in one message. The so called *bulking* enables to reduce the traffic and the overhead that is needed for each single message. The cost model allows predicting the time needed for a synchronization process. Therefore, we are able to determine the latest possible point in time to send the synchronization message for a certain change. Currently bulking of delta messages containing independent changes is considered, i.e. no change will be overwritten. Full-copy update messages and overlapping changes are already examined but not described in this paper.

In (Lenz, 1996) coherency predicates for consistency are introduced. These are *version distance*, *value divergence* and *temporal distance*. The predicates define if a BO replica is still valid after a master was changed. The distance between the BO versions, the differences of the BO content or just a certain amount of time must not exceeded. The bulking of changes results in a delay for synchronization messages. Therefore, the temporal distance Δt for the replicas is crucial. It must not be exceeded to fulfill the consistency constraints.

For simplification we assume that Δt is constant for all BOs. The temporal distance defines the maximum time until a change of a BO has to be incorporated at all replicas. In Fig. 6 three changes of a BO at random time are depicted. The version of the BO changes from V_0 to V_1 to V_2 to V_3. The changes are committed at the times t_1, t_2 and t_3. Once a change was committed the temporal distance for each change must not exceed $t_1 + \Delta t$ for V_1, $t_2 + \Delta t$ for V_2, and $t_3 + \Delta t$ for V_3.

The assembling and disassembling of the message header and the transfer of the overhead is necessary for each message and takes a fixed amount of time T_{Offset} (called offset). In Fig. 6 the bulking of the changes of V_1 and V_2 is depicted. Both changes are included in one synchronization message. Due to the temporal consistency restriction the time $t_1 + \Delta t$ must not exceeded. Therefore, we have to consider the time that is needed to process and transfer the mes-

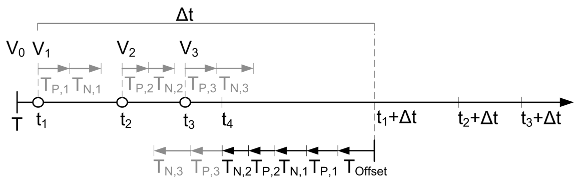

Figure 6: Delay of Synchronization Message.

sage. Additionally to T_{Offset} the times for processing the synchronization message of the first change $T_{P,1}$ as well as the transfer time $T_{N,1}$ are necessary. The times for the second change are $T_{P,2}$ and $T_{N,2}$. An additional offset for the second change is not necessary because it is sent within the same synchronization message. All values can be determined with the help of the cost model. The synchronization message can sent at the latest at $(t1+\Delta t)-(T_{Offset}+T_{P,1}+T_{N,1}+T_{P,2}+T_{N,2})$ which equals t_4.

In the example, the third change at t_3 cannot be included in the same synchronization message. The sum of all processing times $T_{P,1}$, $T_{P,2}$, $T_{P,3}$, all transfer times $T_{N,1}$, $T_{N,2}$, $T_{N,3}$ and the offset exceeds the time left between t_3 and $t_1+\Delta t$. Bulking all three changes in on synchronization message violates the temporal distance Δt for the first change V_1.

8 CONCLUSIONS

This paper discusses an approach for adaptive synchronization of business objects replicated at the middle-tier. We introduced a profiling for BOs and system parameters. Profiling allows determining the processing and transfer costs for the synchronization. The sending of full copies of BOs or delta synchronization messages as well as temporal consistency constraints are considered. A cost model based on an experimental evaluation allows configuring the used replication strategy to achieve an efficient synchronization. A validation was done by profiling real BO instances and the implementation of a simulation environment. The introduced approach of adaptive synchronization is applicable for an initial configuration of the replication strategy for BOs and an adoption during runtime.

REFERENCES

Ameling, M., Roy, M., and Kemme, B. (2008). Replication in service oriented architectures. In Helfert, M., editor, *ICSOFT*, pages 103–110. INSTICC Press.

Ameling, M., Wolf, B., Armendariz-Inigo, J. E., and Schill, A. (2009). A cost model for efficient business object replication. In *AINAW '09 (to appear)*.

Barga, R., Lomet, D., and Weikum, G. (2002). Recovery guarantees for general multi-tier applications. In *ICDE*.

Felber, P. and Narasimhan, P. (2002). Reconciling replication and transactions for the end-to-end reliability of CORBA applications. In *(DOA)*.

Gray, J., Helland, P., O'Neil, P., and Shasha, D. (1996). The dangers of replication and a solution. In *SIGMOD*, pages 173–182.

Killijian, M.-O. and Fabre, J. C. (2000). Implementing a reflective fault-tolerant CORBA system. In *SRDS*.

Lenz, R. (1996). Adaptive distributed data management with weak consistent replicated data. In *SAC '96*, pages 178–185, New York, NY, USA. ACM.

Marta Pati n.-M., Jiménez-Peris, R., Kemme, B., and Alonso, G. (2005). Middle-r: Consistent database replication at the middleware level. *ACM Trans. Comput. Syst.*, 23:375 – 423.

Othman, O., O'Ryan, C., and Schmidt, D. C. (2001). Strategies for CORBA middleware-based load balancing. In *IEEE Distributed Systems Online*. http://www.computer.org/dsonline.

Pacitti, E., Minet, P., and Simon, E. (1999). Fast algorithm for maintaining replica consistency in lazy master replicated databases. In *VLDB*, pages 126–137.

Pedone, F., Guerraoui, R., and Schiper, A. (2003). The database state machine approach. *Distributed and Parallel Databases*, 14(1):71–98.

Perez-Sorrosal, F., Patiño-Martínez, M., Jiménez-Peris, R., and Kemme, B. (2007). Consistent and scalable cache replication for multi-tier j2ee applications. In *Middleware*.

Plattner, C. and Alonso, G. (2004). Ganymed: Scalable replication for transactional web applications. *Middleware*, pages 155 – 174.

Plattner, C., Alonso, G., and T.-Özsu, M. (2007). Extending DBMSs with satellite databases. *The VLDB Journal*.

Salas, J., Perez-Sorrosal, F., Marta Pati n.-M., and Jiménez-Peris, R. (2006). Ws-replication: a framework for highly available web services. *WWW*.

W3C (2002). Web services. http://www.w3.org/2002/ws/.

Wu, H. and Kemme, B. (2005). Fault-tolerance for stateful application servers in the presence of advanced transactions patterns. In *(SRDS)*.