Retrospection on the HERMES project: Implementation of a Heterogeneous Transaction Management System

Aija Palomäki Antoni Wolski Jari Veijalainen Jari Jokiniemi

Technical Research Centre of Finland (VTT)
Laboratory for Information Processing
Lehtisaarentie 2A, 00340 Helsinki, Finland

Internet: <first name>.<last name>@vtt.fi

Abstract

The goal of the HERMES project was to conceive means to integrate existing heterogeneous database products in a transaction management system. The project produced the 2PC Agent method based on the 2PC protocol. The method is based on simulation of the prepared state at participating sites and it guarantees serializable executions in the presence of site and transaction failures. A restriction on the behaviour of local transactions is imposed: the (necessary) condition called DLRP has to be maintained, meaning no local transaction should update the data read by a subtransaction being in the prepared state.

The project experienced a diminishing interest in the part of the industrial supporters, and the prototype software produced was not utilized. Consequently, a question of relevance of the original goals arises.

1: Introduction

The HERMES project was carried out at the Laboratory for Information Processing of the Technical Research Centre of Finland (VTT) during the years 1988—1991. The project was 70 % government-funded, with the rest of the funding provided by the industrial partners participating in the project. The industrial partners — 10 of them altogether — included banks, insurance companies, telecommunications companies, system houses and a machinery manufacturer. The project partners were attracted by the possibility to integrate their dispersed transaction processing systems which were confined to product-specific environments.

We set our goal at no less but to achieve serializable executions of transactions spanning over a set of heterogeneous database products, in the presence of typical failures. We also strove for a minimally restrictive solution, in the sense of performance and also, local autonomy [16]. We discuss the system assumptions, failure types and the objectives in a more detail in Section 2.

The approach known as the *2PC Agent method* was produced and refined during 1989—1991. The final version of the prototype system HERMES became operational in the Spring of 1991. The major results of the work are summarized in Section 3. The ramifications of the results are discussed in Section 4.

2: Premises and objectives

The HERMES system comprises, physically, of the Participating Sites and Coordinating Sites (Fig.1).

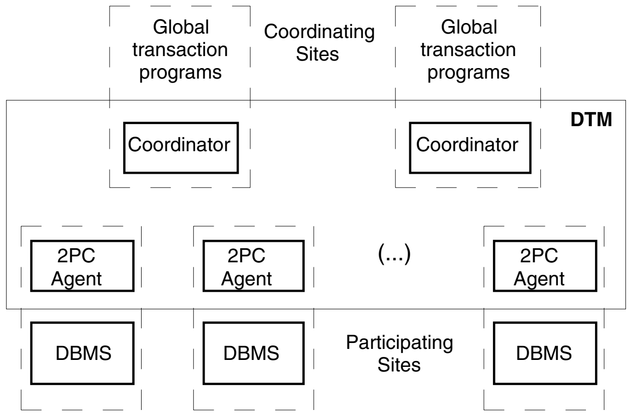

Fig. 1. Software components of a heterogeneous transaction management system.

The goal was to define the Distributed Transaction Manager (DTM), in the sense of [13], having components at all sites.

Each DBMS (database management system) represents a database product. The products may be heterogeneous. The typical characteristics of the contemporary database products are:

- the only way to interact with the DBMS is by using the DML language supported by the product, in our prototype — the SQL;
- they have the one-phase commit interface, i.e. the COMMIT and ROLLBACK commands are provided, but the PREPARE command is not;
- they apply *rigorous* [12] schedulers exemplified by a typical strict two-phase lock (S2PL) manager [4];
- their behaviour cannot be modified.

The two-phase commit protocol (2PC) [14] is used within the DTM. The DTM components at the Coordinating Sites execute standard 2PC coordinator algorithms. As the DBMSs necessarily do not have the prepared state capability, the prepared state has to be simulated within the DTM. This is done in the 2PC Agents.

At the Coordinating Sites, the global transaction programs interact with the DTM using the SQL language. The DTM decomposes the global transactions into subtransactions and passes them to the appropriate Participating Sites. In addition to these subtransactions, there are local transactions originated by programs interacting directly with the DBMSs. The DTM has no knowledge of the local transactions.

The objective is to enforce (view) serializable [4] overall histories (i.e. executions of global and local transactions) in the presence of failures. The failures include:

- Participating Site failures. This involves a total loss of the volatile memory and aborting all the active transactions at a site. Similar are the failures of the transaction management software (DTM, DBMS).
- Transaction failure. A unilateral abort of a transaction, by the DBMS.

The latter failure type has not been dealt with much in the literature. However, in many database products, a transaction is allowed to be aborted by the system at any time—also when no database command is being executed—as long as the transaction has not been committed. The reasons for this behaviour are, usually, implementation-dependent internal resource conflicts, e.g. a circular buffer overflow or an internal table space overflow.

Some researchers question the existence of such a behaviour. On the other hand, the DBMS implementers and users confirm the existence of such transaction failures. We were also able to demonstrate the failure in Ingres v. 6.2 by having one transaction waiting for commit and then overflowing the log buffer with extremely long transactions. Eventually, the system aborted the waiting transaction.

The transaction failure is difficult to deal with because it may happen to a subtransaction being in the prepared state, from the DTM point of view. Our solution to this problem is presented in the next section.

3: Results and related work

When we started our work, only few papers were published in the area. In the opening work of [13], some main problems were identified together with the general DTM architecture plus some elementary solutions. In [1], the use of a log and "retrying failed requests" was proposed. This meant, in fact, the re-execution of transaction programs.

In our approach, the DBMSs are autonomous and the only knowledge the DTM has about the global transaction programs is their SQL commands. Therefore, the DTM can not affect the course of the global transaction execution by any way other than by passing, delaying, rejecting or resubmitting the SQL commands.

During normal operation the DTM only passes the commands to the DBMSs. In case of a subtransaction failure, DTM simply rejects the rest of the subtransactions' SQL commands if the subtransaction has not reached the prepared state yet. Otherwise, the prepared state has to be recovered. We chose to do that by *resubmitting* the SQL commands of failed subtransactions [19]. For this purpose we decided to store the SQL commands in a special log. We also used the idea found in [9], whereby, implementing the log in a local database, the insertion of the commit record into the log may be committed atomically, together with the subtransaction that caused the insertion. The resubmission concept and the simulation of the prepared state were also independently presented in other works dealing with site recovery, e.g. [2, 3, 7, 10 and 11].

Other works at that time dealt with heterogeneous concurrency control issues in failure-free environments. Contrary to that, we decided to utilize a possible homogeneity of concurrency control. In [6], it was shown that, in a system of distributed S2PL schedulers, conflict serializable executions are produced for an arbitrary set of local and global transactions. The result was generalized, independently in [19] and [10], for the class of rigorous schedulers.

The main problem was, however, how to deal with failures jeopardizing the serializability. We understood that if we wanted to guard correctness by rejecting the global transactions possibly offending the serializability, this was to be done before the subtransactions reached the prepared state. We decided to look for potential conflicts at the time the PREPARE command of a subtransaction was processed, and called this step the *prepare certification.*. The *alive time interval based prepare certification* [20] is done by checking the time intervals subtransactions have been *alive* (i.e. not committed or aborted) at a Participating Site. A failure and a subsequent resubmission creates a gap in this time interval. And, if a subtransaction being certified has been alive during that time gap, there is a possibility of a conflict between these two subtransactions. If such an overlap is detected, the prepare certification fails and the PREPARE command is rejected.

In [20], we added a step of *commit certification* to avoid serializability errors caused by indirect conflicts involving local transactions. The role of the commit certification was to force the commits into certain correctness-ensuring order (thus achieving the characteristics known also as the strong recoverability [5] or the commit ordering [15]). A new set of certification algorithms was presented in [17]. The proofs of correctness are in [18]. A special tree-based transaction model was developed to facilitate the proofs.

A significant condition called Denied Local updates for Reread data in the Prepared state (DLRP) was discovered. DLRP means that local transactions are not allowed to update data read by subtransactions being in the simulated prepared state. An important result, for all the resubmission-based approaches, is the following theorem:

Theorem. Necessary condition for conflict serializability (the paraphrased Th. 8 of [18]).

If 1) the overall history is conflict serializable,
2) the local histories are rigorous and
3) no two subtransactions being in the prepared state have conflicting operations,

then the DLRP holds at all sites.

If the DLRP is not maintained, and a subtransaction fails, a local transaction may update an item just read by the failed subtransaction and commit, causing an irrevocable serializability error. For examples of the serializability errors, see e.g. [7] and [17]. Note that subtransaction recovery is performed by resubmission of the SQL commands, with no respect given to the fact that the database state might have changed in the meantime.

In reality, enforcing DLRP is difficult. One solution is to submit also all the local transactions to the DTM. But, this violates the local design autonomy [8], because local transaction programs should be re-programmed. Local data could be divided into distinct locally and globally updateable sets, as proposed in [7], and the transaction programs could be designed accordingly. Instead of the data space, one could divide the data usage time, using administrative or technical means, into distinct local and global update usage time intervals. Or, a certification function call could be added to each local transaction program, which however, violates the local design autonomy as well. In other words, in order to maintain the overall serializability, the local autonomy has to be restricted in one way or another.

At the time of writing, the only known competitive method dealing with transaction failures, is the one of [7]. The comparison of the two approaches shows that they both make use of similar assumptions about the environment, but the solutions are quite different. The approach of [7] utilizes a centralized scheduler, while the 2PCA method is fully distributed.

The core of the method in [7] is the *commit graph*. It is an undirected graph whose nodes are global transactions and Participating Sites. An edge connects a transaction node T_j with a site node S_i iff the global subtransaction $T^i{}_j$ is in the prepared state. The loop in the graph signals a potential conflict among global and local transactions. Thus, the granularity of the potential conflict detection resolution is that of a site. In the 2PC Agent method, the same goal is achieved by means of the prepare and commit certification performed at Participating Sites. The resulting restrictiveness of the methods may vary vastly. It may be shown that, with certain assumptions, the 2PC Agent algorithms do not abort any transactions in a failure-free situations, but the corresponding histories are rejected by the method of [7] because of the site-level granularity in the commit graph. However, an exact restrictiveness assessment requires further study.

On the other hand, the method of [7] is deadlock-free, while the 2PC Agent method requires an additional deadlock detection mechanism (like timeout).

4: Industrial feedback

During the HERMES project, the interest of our industrial partners in the project was declining over time. In the end, it turned out that there was not anyone eager to pilot test the approach in a real environment, not to speak of making it a product. So, obviously we got something wrong in the requirements analysis phase.

Actually, our partners rarely used ACID transactions, even in their centralized database systems. In heavy-load systems, most of the ACID characteristics, namely serializability, atomicity and isolation were often compromised in favour of performance. The long-term correctness and failure resilience of the application systems relied on guarding the semantic correctness of transactions within the application code. This meant, e.g. utilizing commutativity of operations at the application level for concurrency control, and compensation programs for recovery purposes (especially in banks). A proliferation of application-level logs and various consistency checking programs underlined this approach.

We also got the impression that, after all, our partners did not feel secure about distributing transactions across physical system boundaries. Instead other interoperability means (e.g. data copying) were substituted. Furthermore, the enterprises did not seem to want to use commercially unsupported systems software; equally the role of in-house development in this area was diminishing.

5: Conclusions

The HERMES project established a way to integrate existing heterogeneous database products in a transaction management system. It produced the 2PC Agent method which guarantees serializable executions of global transactions in the presence of site and transaction failures but imposes a restriction on the scheduling of local transactions. The experience of the project indicated that semantic-based solutions to transaction management are worth pursuing, at the DTM level, in order to preserve the local autonomy. On the other hand, if the autonomy requirements can be relaxed, serializability-based approaches are more feasible. The appeal of the serializability lies in the unbeatable simplicity of the concept, which leads to efficient and reliable application programming.

References

[1] R. Alonso, H. Garcia-Molina and K. Salem, "Concurrency Control and Recovery for Global Procedures in Federated Database Systems", *Quarterly Bull. IEEE Tech. Comm. on Database Engineering*, Vol. 10, No. 3 (1987), pp. 5 - 11.

[2] K. Barker, "Transaction Management on Multidatabase Systems", TR 90-23 (Ph.D. thesis), August 1990, Dept . of Computing Science, The Univ. of Alberta, Edmonton, Alberta, Canada.

[3] K. Barker and M.T. Özsu, "Reliable Transaction Execution in Multidatabase Systems", *Proc. First Internat.Workshop on Interoperability in Multidatabase Systems* (Kyoto, April 7-9, 1991), pp. 344-347.

[4] P.A. Bernstein, V. Hadzilacos and N. Goodman, "Concurrency control and recovery in database systems", Addison-Wesley Publ. Comp., 1987.

[5] Y. Breitbart, D. Georgakopoulos, M. Rusinkiewicz and A. Silberschatz, "On Rigorous Transaction Scheduling", *IEEE Trans. on Software Eng.*, Vol. 17, No. 9 (1991), pp. 954 - 960.

[6] Y. Breitbart and A. Silberschatz, "Multidatabase Systems with a Decentralized Concurrency Control Scheme", *IEEE Distributed Processing Tech. Comm. Newsletter*, vol. 10, no. 2 (1988), pp. 35-41.

[7] Y. Breitbart, A. Silberschatz and G.R. Thompson, "Reliable Transaction Management in a Multidatabase System", *Proc. 1990 ACM SIGMOD Conf.* (Atlantic City, 23 - 25 May, 1990), pp. 215 - 224.

[8] F. Eliassen and J. Veijalainen, "Language support for Multi-database Transactions in a Cooperative, Autonomous Environment", *Proc. TENCON 87* (Seoul, 25-28 August, 1987), pp. 277 - 281.

[9] H. Garcia-Molina and K. Salem, "Sagas", *Proc. 1987 ACM SIGMOD Conf.* (San Francisco, 27 - 29 May), pp. 249 - 251.

[10] D. Georgakopoulos, "Transaction Management in Multidatabase Systems" (Ph.D. thesis), December 1990, Dept. of Computer Science, University of Houston, Houston, Texas.

[11] D. Georgakopoulos, "Multidatabase Recoverability and Recovery", *Proc. First Internat. Workshop on Interoperability in Multidatabase Systems* (Kyoto, April 7-9, 1991), pp. 348-355.

[12] D. Georgakopoulos, M. Rusinkiewicz and A. Sheth, "On Serializability of Multidatabase Transactions through Forced Local Conflict", *Proc. 7th Conf. Data Engineering* (Kobe, 8-12 April, 1991), pp. 314-323.

[13] V. Gligor and R. Popescu-Zeletin, "Transaction Management in Distributed Heterogeneous Database Management Systems", *Information Systems*, Vol. 11, No. 4 (1986), pp. 287 - 297.

[14] ISO/IEC 9804: 1990, "Information technology—Open Systems Interconnection—Service definition for the Commitment, Concurrency and Recovery service element", Internat. standard, ISO/IEC, Geneva 1990.

[15] Y. Raz, "The principle of Commit Ordering, or Guaranteeing Serializability in a Heterogeneous Environment of Multiple Autonomous Resource-Managers", *Proc. 18th VLDB Conf.* (Vancouver, August 23-27, 1992).

[16] J.Veijalainen, "Transaction Concepts in Autonomous Database Environments", GMD-Bericht Nr. 183 (Ph.D. thesis), R. Oldenbourg Verlag, Munich, Germany 1990 .

[17] J. Veijalainen and A. Wolski, "Prepare and Commit Certification for Decentralized Transaction Management in Rigorous Heterogeneous Multidatabases", *Proc. Eighth Internat. Conf. on Data Engineering* (Tempe, Feb. 3-7, 1992), pp. 470-479.

[18] J. Veijalainen and A. Wolski, "The 2PCA Agent Method for Transaction Management in Heterogeneous Multidatabases, and its Correctness", Research Report No. J-10, June 1992, Helsinki, VTT, Laboratory for Information Processing.

[19] A. Wolski and J. Veijalainen, "2PC Agent Method: Achieving Serializability in Presence of Failures in a Heterogeneous Multidatabase", *Proc. IEEE PARBASE-90 Conf.* (Miami Beach, 7-9 March, 1990) pp. 321 - 330.

[20] A. Wolski and J. Veijalainen, "2PC Agent Method: Achieving Serializability in Presence of Failures in a Heterogeneous Multidatabase", In: Rishe, N., Navathe S., Tal, D. (eds.). *Databases: Theory, Design and Applications.* IEEE Computer Society Press, 1991, pp. 268-287.