JADE implementation of
Heterogeneous Nested Transactions

Matías Alvarado1 & Luis A. Gama-Moreno2,3
matiasa@imp.mx, lgama@sagitario.cic.ipn.mx

1PIMAyC, Instituto Mexicano del Petróleo
Eje Central L. Cárdenas 152, CP 07730, México, D.F.
2Instituto Tecnológico de Zacatepec
Calzada Tecnológico S/N, Zacatepec, Mor. C.P. 62780, México
3CIC, Instituto Politécnico Nacional

Abstract. A mechanism to reduce the amount of lost work-done using Nested Transactions (NT) and Autonomous Agents is introduced in this paper. NT provides a greater failure-tolerance model than the Distributed Transactions because a NT can be divided into alternative sub-transactions, such that each can commit independently. On the other hand, Autonomous Agents are able to process sub-transactions while maintaining an intense but consistent communication –critical in distributed systems.

1 Introduction

In Distributed Transaction (DT), data availability depends on its location, that is the reason why the processing of one transaction splits into sub-transactions1, in the different storage-data-nodes. The communication between the nodes is subject to network failures. If so, successful processed transactions frequently must be aborted to preserve the ACID (Atomicity, Consistency, Isolation and Durability) properties of a global transaction [15]. The control of a global transaction resides in the initiating node of the transaction or coordinator. Each logic-unit of work processed in each node can commit or abort independently, according to confirmation rules in 2PC (Two-Phase Commit) protocol. Nevertheless, the disadvantage of using DT is that the work of the transactions that have been successful committed, must be aborted if at least one sub-transaction fails.

Nested Transactions (NT) model provides a greater failure-tolerance [14]. A NT allows the division of a transaction into sub-transactions –like a DT, creating a hierarchical transactions tree. The main advantage of the NT is that it is possible to commit the global

---

1 A sub-transaction is a transaction itself. So, sub-transaction term are replaced just by transaction.
transaction with those successful sub-transactions that have fulfilled their objectives even though there are sub-transactions that have been aborted [9], [14].

Another important aspect is the constant communication that the nodes involved in the processing of DT or NT must maintain. With the use of Software Agents that functionality is obtained, due to their characteristics of autonomous processing and interaction capacity; also they are able to provide oriented services to the sub-transaction processing with the purpose of reaching their objectives, either commit or abort the data of the transaction assigned to the agents. Likewise, the executing agents on JADE (Java Agent DEvelopment) platform have the required coordination mechanisms for the Distributed and Nested Transactions processing. On the other hand, the nodes that support the transactions processing are usually heterogeneous: different hardware architectures and operating systems or DBMS (Data Base Management System). This heterogeneity can be supported adequately by Multi-Agent System (MAS), due to autonomy capacities, interaction, services supplying and goals achievement, characteristics of agents [3].

The rest of this paper is organized as follows. Section 2 describes technologies used in the implementation of Heterogeneous NT (HNT). Section 3 lists the roles carried out by an agent in HNT processing. Section 4 is about the advantages of using agents for HNT processing. In Section 5 a case of study, using software agents running over JADE platform is outlined. The case of study includes transactions processing in oil platforms of different locations. Finally, in Section 6 conclusions are presented.

2 Heterogeneous Nested Transactions and Autonomous Agents

Next section describes Nested Transactions model, software agents and JADE platform technologies for MAS development.

2.1 Nested Transactions

As has been said, NT provides a mechanism with greater failure-tolerance than DT: it can also splits the transaction into logic-units or sub-transactions as in DT, the failure of a transaction does not entail the abortion of other transactions. In this approach, the initial transaction or root is able to commit just with the work confirmed by some of its transactions as long as the objectives are satisfied within the transaction. If some of the transactions presents failures it is aborted but not necessary the whole transaction –as is the case for DT.

The NT model is a solid alternative for applications on mobile devices prone to frequent failures [2], [3]. Fig. 1 shows a NT hierarchical tree: the top-level transaction is denominated root, the transactions that have transactions are called parents and their transactions are called children. Transactions without sub-transactions are called leaves, and they are generally able to do read/write operations. Each leaf represents a flat
transaction, in other words, indivisible into logic-units of work. Thus NT advantages are: the global transaction can be divided into sub-transactions forming a hierarchy of units of work controlled by the global transaction. Each sub-transaction can be executed independently from the others (and the final commitment depends on the global transaction). Processing distribution among nodes reduces dependency within transactions; the NT commitment is possible even though some of its transactions have aborted.

Fig. 1. Nested Transaction example.

2.2 Autonomous Agents

Autonomous Agents (AA) are software entities able to execute tasks in a flexible and autonomous way [12]. In addition, to reach their objectives they communicate with other agents of the system through a communication language such as FIPA-ACL [6]. An agent has the following characteristics:
• **Autonomy**: execution capacity without extern intervention and with own control on its state; they are only affected by incoming messages, which are processed internally.

• **Concurrency**: capacity of independent execution from other agents.

• **Reactive**: they respond to environment events.

• **Proactive**: they plan and act in order to reach their own goals.

• **Knowledge**: it is the information that they have about environment, other agents and the system state in general. They interchange knowledge through communications and interactions.

• **Reasoning**: they have a mental structure given by internal inference mechanisms to manage their knowledge.

• **Service**: they are actions that an agent can achieve; some are complementary to those of other agents to solve problems that go beyond their own capacities.

• **Interaction**: it is set by interchanging communicative messages between agents of a MAS, or through physical actions.

These constitutive elements of agents are specified by means of ontology, a particular one for each type of agent: as for relations between agents. Ontology facilitates message interchange and understanding of the content. If two agents use the same ontology, then interoperability is facilitated in heterogeneous applications, which respectively support the agents. Ontology is about knowledge and service. Those of knowledge describe domains as well as propositions (facts) that characterize the system. Service ontology helps to describe and homogenize services: the point is that agents can reason about the services of other agents to determine their use.

2.3 JADE platform

JADE is an agent development platform [3], [11] implemented in Java language. JADE uses Java capacities such as: **Java RMI (Remote Method Invocation)**, **Java CORBA IDL (Common Object Request Broker Architecture Interface Definition Language)**, **Java Serialization** and **Java Reflection API** for the construction of distributed applications. JADE simplifies Multi-Agent Systems (MAS) implementation through a middle-ware compatible with FIPA (Foundation for Intelligent Physical Agents) specifications. The JADE platform can be distributed through different machines (maybe with different operating systems) the configuration can be controlled by a remote graphic user interface. The communications architecture in JADE offers a flexible and efficient message mechanism.
FIPA communications model has been implemented and its components have been clearly defined and integrated: interaction protocols, envelops, ACL (Agent Communication Language), languages of content, codification schemes, ontologies and transportation protocols. Many interaction protocols defined by FIPA are available [4], [6].

3  Heterogeneous NT with Autonomous Agents

As mentioned before, processed data by a transaction can be placed on different locations (nodes), and in many cases, manipulated by different DBMS. For example, if the temperature processing of diverse oil platforms is required for statistical purposes, this processing can be carried out through a NT schema in order to improve the failure-tolerance. Likewise, various DBMS can be involved due to particular local nodes requirements. Nevertheless, the transaction is processed like a single one, without concern the data location and the used DBMS.

3.1 Roles of Autonomous Agents

There are three different roles in this scenario: User, Coordinator and Executor, defined as follows:

• User Agent (UA): it represents a user application that requires transaction processing. The UA is in charge of initiating the transaction by sending a message to CA.

• Coordinator Agent (CA): it operates on a node that conducts control operations of a transaction, receives a set of instructions and distributes them between the agents that provide the service required for each instruction.

• Executor Agent (EA): it operates on a node (named leaf node), which processes flat transactions; this agent is able to execute requested tasks from CA, also, connect with a DBMS, initiate the transaction, process the required instructions and send a commit/rollback message to CA.

While running on JADE, those executing agents interchange messages about the state of each transaction. Involved agents, register a service specifying the location of the data they are able to process. By means of a message, an AC requests a service to process data in a specified location (i.e. data stored in a platform located at Veracruz). An EA that provides such service, contacts with CA applicant, initiating a parent-child relationship according to the NT scheme described in Section 2.1. EA sends a message to CA depending on the processed transaction state, either a commit or a rollback.
3.2 Heterogeneous NT: creation, ACID properties and termination

Information located at different network nodes, is typically managed by heterogeneous DBMS. Nevertheless, the distributed information management in such system is transparent to users. In the example, in which the average temperature of different oil wells is calculated, using NT, the root transaction (processed by CA), is divided into sub-transactions each one processed by an EA. Each EA keeps information about: DBMS connection, database name, login and password, required for the assigned transaction processing. With the kept information, an EA is able to process a transaction independently of the physical data location. This has implications on ACID properties and the ending of a transaction.

To preserve ACID properties in a transaction, it is critical to guarantee the data consistency in all DBMS [9]. Using NT model [14], only the global or root transaction is the one that keeps ACID properties. The sub-transactions in charge of data processing only fulfill ACI properties but not Durability. Sub-transactions will be durable only if root transaction succeeds. If root transaction aborts, all sub-transactions abort as well. This commit/abort schema is called Closed NT. Nevertheless, if a sub-transaction is not able to commit, it causes that objects remain locked for long periods. In [8] an Opened NT (ONT) is proposed, which consists on letting sub-transactions to commit or abort, independently to other sub-transactions state. In ONT if root transaction aborts, then the work committed by sub-transactions will be compensated.

NT commitment in MAS requires messages passage between the processing node and the coordinator node. The passing of messages between the agents is given by the JADE platform. When an EA is ready to finish, its transaction executes a commit event and sends a message to its CA in order to be registered as a committed transaction. The same is true for abortion.

4 HNT on Autonomous Agents

4.1 Advantages

The autonomy of agents and, in general, of the MAS architecture, provides diverse advantages in HNT processing. Each agent provides access to data stored in a given oil platform. In addition, each agent is able to execute locally without strict dependency to (relative autonomy) a central system. With this approach, monitoring processes on any nodes are possible, also, monitoring different transactions in a decentralized way without the need of a specialized node –which presents greater degree of vulnerability. These characteristics give at the same time the following advantages:
1. greater transactions concurrency,
2. relaxing isolation level between transactions,
3. improving applications throughput,
4. commitment/abortion of each transaction independently to other transactions state,
5. messages sending with semantic information about content and codification language.

MAS improves concurrency of transactions, because each agent executes a transaction locally in a given node, and its autonomy reduces dependency between transactions. This avoids locked-object latency, and reduces deadlocks and cascade-aborts. Consequently, throughput of applications is enhanced. On the other hand, the agent that supports the Nested Transaction reaches its goals in spite of other transactions failures. With content semantic definition and messages coding, agents can interact with others to reach their goals.

### 4.2 HNT on mobile devices using agents

The advantages of data distribution in heterogeneous DBMS and processing distribution using Autonomous Agents are a robust alternative for environments characterized by its instability and frequent failures. So, it is the case for wireless devices (PDAs, laptops and cell phones) whose basic characteristic is the mobility [7], [16], [18]. In the so-called mobile computing, processing problems are generated by changes in cell areas, limited battery energy, little storage space, small display and most of all frequent often announced disconnections cases [2], [5], [13], [17].

Mobile agents are an excellent paradigm for HNT implementation on mobile devices since they are able to support disconnected operation in several ways:

- They jump from a device that experiences frequent disconnections and reconnections (a partially connected device) to another one, and return to it even though change their address during reconnection.
- They communicate effectively with other agents to maintain a decentralized control of the state of the transaction assigned to each mobile agent.
- They process a transaction locally while the device remains disconnected.
- They move programmers away from rigid client/server model to the more flexible peer-to-peer model, in which process communicate as peers and act either as clients or servers.

Also, mobile agents can create scalable applications due to their capacity to move the process where the data are located, instead of bringing the data to where it is required for processing. This is an ideal characteristic for HNT processing with mobile devices [10].
5 A case of study

In a region, with multiple oil platforms, a data scenario is distributed on different DBMS, with local information corresponding to each platform. Consider the following execution of a transaction:

```
BEGIN TRANSACTION X
  UPDATE well
  SET presion = 0
  WHERE ubicacion = 'VERACRUZ'
      OR ubicacion = 'MEXICO';
  UPDATE PLATFORMS
  SET presion = 1;
COMMIT
```

Fig. 2. Example of a transaction.

The code of Fig. 2 shows an example of a transaction that can be processed by means of a Nested Transaction. The instructions are verified by the coordinator to decide whether they are distributed between the different nodes. For example, instruction `UPDATE pozo
WHERE ubicacion = 'VERACRUZ' OR ubicacion = 'MEXICO',` is separated between nodes that offer those services. Fig. 3, shows the final operations distribution. The coordinator node receives the transaction, and requests the services named “well” and “platform”. The next instruction is assigned to Node 1:

```
UPDATE well
SET presion = 0
WHERE ubicacion = 'VERACRUZ'
      OR ubicacion = 'MEXICO';
```

which is divided into two instructions depending on their location. Therefore, \textit{leaf1} receives the instruction to process the data of the platform located in MEXICO using the SQL Server DBMS:

```
UPDATE well
SET presion = 0
WHERE ubicacion = 'MEXICO';
```

\textit{Leaf2} performs the instruction for the data located in VERACRUZ using the Borland Interbase DBMS.

```
UPDATE well
SET presion = 0
WHERE ubicacion = 'VERACRUZ';
```

Finally, the instruction that processes “platforms” information (without specifying location) is alternated to \textit{leaf3}, using Oracle (see Figure 3).

The CA performs a recognition of the received message and depending on its language coding – in ANSI-SQL92 [1], it will decide to request access services to tables, i.e. agents
designated for the transactions processing. These agents denominated, “node x”, check the SQL instructions received to determine if they split into sub-transactions, or they make the transaction like an EA. In case of dividing themselves, they became coordinator, therefore, they receive the messages from leaves transaction, and decide either to confirm or abort. Each EA performs a flat transaction with the instructions received by their parent transaction, in a specific DBMS.

Fig. 3. Architecture of the case of study: Heterogeneous Nested Transactions.

6 Conclusions

Using a Nested Transactions model and Autonomous Agents enhances performance and failure-tolerance of applications, especially focused upon transactions that run in mobile devices. Applications running with agents (in a MAS) on JADE platform make data location transparent. The JADE communications mechanism supports the coordination between agents, which execute transactions, in this work the coordination is about UA, CA and EA. Each transaction processed by an Autonomous Agent controls its commitment/abortion independently of the rest of the transactions. Other advantages of using MAS are, greater transaction concurrency due to the relaxation of the independence
level, and consequently greater data availability bringing up better applications throughput. Such characteristics are specially adapted to transaction processing supported by mobile devices whose relevance is its ubiquity.

References