THE PERSEUS SAGA

LA SAGA PERSEUS

Luciano García-Bañuelos1, Christine Collet1
lgarcia@imag.fr / Christine.Collet@imag.fr

Recibido: noviembre 12, 2003 / Aceptado: enero 30, 2004 / Publicado: febrero 18, 2004

ABSTRACT. We present PERSEUS, a software framework allowing to build customized persistent object managers. Three types of manager can be built: unreliable, reliable based on checkpoints and reliable based on transactions. Such configurations are possible because of the architecture and the multi-level recovery used in PERSEUS. This recovery method allows to separate system fault tolerance from transaction support, which are usually implemented together.

KEYWORDS: computer science, software framework, object managers, database management systems.

RESUMEN. Presentamos PERSEUS, un marco computacional que permite crear administradores de objetos persistentes a la medida. Se pueden crear tres tipos de administradores: no confiables, confiables basados en puntos de verificación, y confiables basados en transacciones. Dichas configuraciones son posibles debido a la arquitectura y la recuperación multi-nivel utilizados en PERSEUS. Este método de recuperación permite separar la tolerancia de fallas del sistema del soporte a transacciones, los cuales generalmente son implementados de manera conjunta.

PALABRAS CLAVE: Computación, marco computacional, administradores de objetos, sistemas de administración de bases de datos

Introduction

The management of large amounts of persistent data has attracted considerable research interest during the last decades. Several research communities have proposed convenient solutions in terms of both performance and reliability. Database Management Systems (DBMSs) can be considered as the more representative solution available for persistence management. At the heart of information systems, DBMS technology has been evolving continuously to meet new application requirements.

We contend that current DBMS architectural approaches are not flexible enough to accommodate promptly the technology required. Our argument is that a DBMS is usually developed, packaged and deployed with a fixed set of functionalities, which could mismatch the target application requirements. For instance, if the target application environment is resource-limited, a full-fledged DBMS may be cumbersome. Another possibility is that the DBMS lacks for a feature, but it does not provide a way to accommodate any new feature. In both cases, the solution is usually to develop a new ad hoc infrastructure.

The objective of our work is to design and develop an adaptable infrastructure for persistence management. We have taken well-known techniques, characterizing architecture and design patterns in order to develop an adaptable infrastructure. The result is a software framework called PERSEUS, which allows to build a family of persistence managers. The problems that we have addressed are the characterization of persistence management, the identification of the relationships between persistence and other system concerns, and the design of an adaptable infrastructure flexible enough to accommodate different approaches for fault tolerance. The evolution of this work can be tracked in [14].

1 Laboratoire LSR/IMAG, France -www-lsr.imag.fr
Related work

Extensible DBMS

By the end of the 80's, the industrial and academic research communities started efforts for developing what was called the extensible database management systems. In the following, we give a general classification of those approaches.

- **DBMS generators** - The idea of generating some parts of the DBMS has been explored by some researchers. Examples of this approach include the Genesis project [5] and the Volcano query optimizer generator [6]. This approach has been apparently abandoned, presumably because of the difficulties due to the intrinsic complexity of the DBMS software.

- **DBMS with an extensible type subsystem** - Some researchers studied ways to let users extend DBMS type system (i.e. ADT), and underlying access methods (i.e. indexes). Prototypes such as doctoral studies financed by the CONACyT Scholarship program of the Mexican government Postgres [7] at the University of Berkeley, and Starburst [8] at IBM are examples of this approach. Note that this approach has effectively evolved, and it is at the origin of the so called object-relational DBMSs.

- **DBMS kernel toolkits** - This approach has been taken in the Exodus [9], Shore [10] and DASDBS [10] projects. The idea was to build a minimal (core functionality) DBMS kernel along with a set of tools allowing to build ad hoc DBMSs. Open OODB [12] is another example of this approach. In contrast to other works, Open OODB focused on the provision of high level system modules such as query processing, distribution and replication. These projects have been testbed for lots of research developments. Note however that these prototypes were almost full-fledged DBMSs, so that lighter weight DBMS implementations could not be envisioned.

Persistence service standards

Among the proposals for persistence service standards, we can mention three of the most widely accepted: the Object Management Group's Persistent State Service (PSS 2.0) [13], the Object Data Management Group standard (ODMG 3.0) [14], and the Java Data Objects (JDO) [15] at the Java Community Process. All of them attempt to provide a standard approach to integrate existing persistence technology (e.g. DBMS) by using a single homogeneous set of interfaces. Effectively, they intend to provide a framework for building persistent applications independently of the underlying support, thus promoting in some scale multi-vendor interoperability.

The three proposals provide a transaction-based processing mode, except for PSS which also provides a non transactional approach. The proposals promote high level APIs (Application Programming Interfaces) hiding implementation details. JDO goes beyond revealing some system internals. Among those details, there are notably APIs to provide hints for cache replacement, hints for swizzling/object faulting scopes, for selection of concurrency control strategies (e.g. locking-based or optimistic approaches), etc.

The OMG has taken from the beginning a service oriented approach. Hence, they proposed the following persistent related services [16]: transactions, querying, and concurrency control. However, although PSS
specifies interaction with the transaction service, nothing is said about interactions with querying and concurrency control services.

The Java Community Process has started defining a generic cache service [17]. Note that the definition process is led by Oracle, so that a DBMS-like approach can be expected. However, there is no up-to-date information about interaction between JDO and the cited cache service.

Our position

Earlier works have kept in mind the goal of building more and more complex DBMS. We argue, however that the process inverse has to be done, trying to breaking down DBMSs in simpler but meaningful software components. Our vision is close to the RISC-style data management blocks described in [18]. Thus, instead of talking about an extensible DBMS kernel, we go beyond by proposing fine-grain software components for mechanisms such as caching, logging and recovery. We have tracked the development of standards for persistence, and other related services. We have witnessed, unfortunately, the lack for integration between those services. We are convinced that the understanding of the subtle dependencies inside the DBMS machinery can be useful to define well integrated services.

PERSEUS overview

This service provides a general purpose logging facility. The major responsibilities of the component that provides this service are the implementation of reliable storage, log record organization, and housekeeping activities such as log truncation. Although the design of this component targets primarily the recovery methods in PERSEUS, it can be used in other contexts such as execution tracing. In order to keep generality, when this component is used in recovery methods, checkpointing and recovery methods must be implemented elsewhere.

Concurrency Control

This component provides an interface allowing to accommodate a large range of concurrency control methods. Furthermore, in the case of locking-based implementations this component can be used in both transactional and non-transactional contexts.

The second group of PERSEUS components, referred as persistence components, provides three types of persistence managers. These components use instances of the base components and implement the underlying management methods. The design of this sub-set of components is based on a multi-layer architecture. As said before, PERSEUS can instantiate three types of persistence managers:

Unreliable

This type of manager does not take into account fault tolerance-related code. Thus, it is intended to be used in contexts where system failures do not have negative impacts data integrity. This is the case of read only systems. Note that updates are not precluded, but the system does not provide any guarantee in face of failures. The component that implements this type of manager deals with the following concerns: object fault and update propagation.
Reliable based on checkpointing

This type of manager provides a first approach for fault tolerance, based on system checkpointing. It provides an operation which allows to explicitly request a checkpoint on the system state, i.e. the current state of the whole set of storage objects. An example of an application that can use this type of persistence manager is that of persistent virtual worlds. In case of failure, the component that provides this type of manager is able to recover a consistent state based on the checkpoint information that has been stored before. The underlying recovery method is based on a redo-only approach. Note that this manager exploits the service provided by the unreliable persistence manager. This allows capitalizing code reuse.

Reliable based on transactions

This type of manager adds the support for ACID transactions. The interest of a persistence manager with support of transactions is obvious. Transactions are a convenient abstraction that facilitates the development of applications with a well defined model of reliability. This manager exploits the redo-only recovery method provided by the reliable persistence manager based on checkpointing to implement transaction durability. Thus, the component that implements this transactional manager is responsible only for transaction atomicity and isolation. To this end, it uses a multi-level recovery method which allows a separation of durability and atomicity supports.

Figure 1. PERSEUS usage scenario
Figure 1. Illustrates one usage scenario of PERSEUS. In such an example, a system programmer is responsible for building a persistence manager based on a provided specification, e.g. PSS. Such a specification provides the definition of the persistence management requirements in terms of data, transactional and access models. Based on the specification and according to the PERSEUS provided component interfaces, the programmer is able to build the underlying service implementation. Thus, a persistence manager results from the instantiation of some PERSEUS provided components cooperating with the code implemented by the programmer.

Results

We developed a first prototype of our infrastructure [4]. We chose the Java language to exploit particularly the high level of portability of its bytecodes. We are interested in using our infrastructure in different configurations of hardware and operating system. The prototype provides a stand-alone storage manager.

Further, another storage manager has been built on top of full-fledged DBMSs using a open source database adapter [19].

We tested a reliable non-transactional configuration of our infrastructure in the context of the PING project [20]. This project aims at providing a platform for building large-scale massively multi-user software, such as large-scale simulation software and networked games. We concluded that a transaction-oriented DBMS is ill-suited, justifying the use of our infrastructure. This configuration has been also tested in handheld computers [3].

The transactional configuration has been tested as a stand-alone persistence service and as a middleware for accessing to data on DBMSs [3]. Furthermore, we developed a partial implementation of the JDO specification. We compared the performances of the PERSEUS-based JDO implementation with those of a commercial product. To this end, we have ported the OO7 database benchmark [21]. We found that in spite of the high modularity of PERSEUS, the underlying implementation provides performances similar to those of the commercial product. The reader is referred to [4] to find more details on the benchmark results.

Conclusions and ongoing work

We have presented the first results of our works looking at unbundling DBMS, and particularly in the definition of an infrastructure for persistence management. Our efforts have resulted in the formulation of architecture principles, component definition, and characterization of component interactions. Our infrastructure can be used to build three types of persistence managers each with a particular fault tolerance approach: unreliable, reliable based on checkpointing and transactional.

By the time of the writing of this document, we work with France Telecom R&D in the re-engineering of PERSEUS. This effort aims at the development of an industrial-strength implementation of PERSEUS and its integration on the ObjectWeb initiative [22]. The ObjectWeb Consortium is an open source software community created by the end of 1999 by France Telecom R&D, Bull and INRIA. Its goal is the development of open source distributed middleware, in the form of flexible and adaptable components. PERSEUS is used for the development of other ObjectWeb software, such as a JDO implementation. To this end, PERSEUS is used along with JORM [23]. This latter is a software framework for the construction of
storage managers with object-based access on top of full-fledged DBMSs. It has to be pointed out that JORM will be used to generate a PERSEUS compliant storage manager.

Furthermore, we are working in the integration of our infrastructure to other services defined within our research project. We are interested in the short term in the development of distributed persistence managers with replica management capabilities. Finally, we continue to investigate the particular data management requirements of large-scale shared virtual worlds and other emerging application domains such as mobile systems.

References