CoDIMS - A Middleware System to Support Visualization Applications in a Grid

Fábio Porto, Gilson A. Giraldi, Jauvane C. De Oliveira,
Bruno Schulze, Vinícius F. Vieira da Silva

Department of Computer Science[1]
National Laboratory for Scientific Computing
Petrópolis, RJ, Brazil

Abstract

In this paper we propose a middleware infrastructure adapted for supporting scientific visualization applications over a Grid environment. We instantiate a middleware system from CoDIMS, which is an environment for the generation of Configurable Data integration Middleware Systems. CoDIMS adaptive architecture is based on the integration of special components managed by a control module that executes users workflows. We exemplify our proposal with a middleware system generated for computing particles trajectories within the constraints imposed by a Grid architecture.

1. Introduction

CoDIMS (Configurable Data Integration Middleware System) [1] is a middleware environment for the generation of adaptable and configurable data integration middleware systems. Data integration systems [2,3,4,5,6,7,8,9] were designed to provide an integrated global view of data and programs published by heterogeneous and distributed data sources. Applications benefit from these type of systems by transparently accessing published resources independently of their localization, data model and original data structure.

With CoDIMS we aim at providing an architecture that can be adapted to new data integration application requirements, so that light weight middleware systems can be generated that conform to the requirements of the applications.

In this paper we focus on the configuration of the CoDIMS environment to support the execution of scientific applications in a grid architecture. As an example of such applications, we consider the computation of particles (i.e. virtual particles) trajectories for a flow vizualization application. The trajectory computing problem (TCP) combines the processing of huges datasets, storing particles and velocity information in a time instance, with intensive calculations for the interpolation of particles positioning in the path.

Our goal is to devise a middleware system to be used in experimentations with different strategies for the efficient computation of particles trajectory in a flow through the use of a Grid [10] infrastructure being developed at the *LNCC* (http://netra01.lncc.br). We introduce the initial design of such architecture and sketch a possible parallel execution engine.

The rest of this paper is organized as follows. In Section 2 we introduce some background in data integration and on the fluid dynamics problem. Next, we in Section 3 we present the CoDIMS approach and its architecture. Section 4 presents the particles tracing problem and specify the middleware components that support its computation in a grid environment. Finally, Section 5 concludes and points to future work.

2. Background and Related Work

According to [11] and [12], database research groups must explore extensibility and componentization in systems development, in order to generate efficient, flexible, and lightweight systems. In the first half of the nineties, some important projects proposed techniques to produce extensible DBMS. EXODUS [13]

and GENESIS [14] were the first projects that proposed a more general approach for DBMS construction through DBMS generators.

More recently, the development of software based on component [15] [16] has gained importance as a technique for developing flexible systems. New application requirements can be served by modifying existing components or by adding new ones with compatible interfaces.

The problem of integrating heterogeneous data sources has received a great attention from the database community. This can be noted by the number of systems proposed, each one having a special type of application in view. One may want to classify these initiatives according to the type of application they were designed for supporting. A first category includes those systems specificaly designed to integrate data from different sources that are the basis for the creation and maintenance of Web sites, like Araneus [2] and Strudel [3]. A second category groups those systems based on mediation technique, like: TSIMMIS [4] and DISCO [5]. A third category are database oriented: MIRO-Web [7], Garlic [8]. Finally, MOCHA [9] and LeSelect[6] are considered data integration middlewares, in the sense that they extend tradional data integration services with mechanisms for user defined function execution. In this scenario, user queries can be seen as simplified workflows over distributed and heterogeneous data and programs.

As suggested by the above classification, these systems are designed having a specific application in mind. Supporting new application requirements may entail quite a strong developing effort. Systems directed towards supporting a wide spectrum of applications are commonly large and heavy which translates into execution inefficiency and complexity in use.

The middleware been proposed aims at addressing scientific vizualizatoin problems. As an example of such applications we will consider particles tracing problems [17,18,19,20]. Such problem can be mathematically defined by an initial value problem [17]:

$$dx/dt = F(x,t),\ x(0)=P_{0,} \quad (1)$$

where $F: R^3 x\ R_+ \rightarrow R$, is a time-dependent vector field (velocity, for example). The solution of problem (1) for a set of initial conditions gives a set of integral curves which can be interpreted as the trajectory of massless particles upon the flow defined by the field $F(x,t)$. Other particle tracing methods can be used (streamlines, streaklines, etc.) through slight modifications of the above equation [17].
The problem (1) in general does not have analytical solution. Thus numerical methods must be used [21]. These methods basically generate the particle trajectory step-by-step following some scheme for field interpolation inside cells[19].

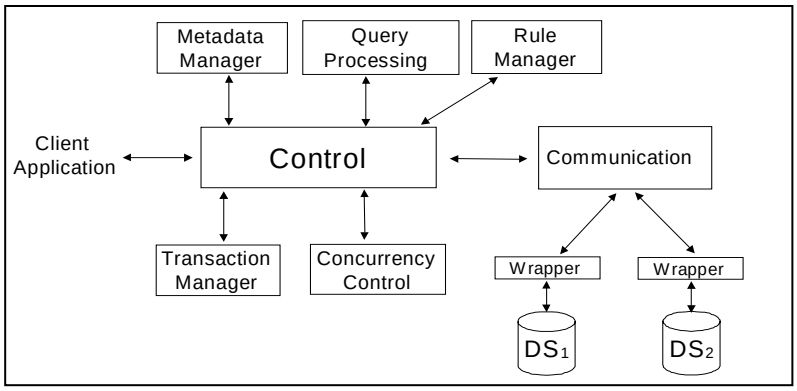

Figura 1 CoDIMS: architecture overview

3. The CoDIMS Approach Overview

In this section, we give an overview of the CoDIMS approach. CoDIMS is a flexible and configurable environment to generate middleware systems. The configuration is obtained through a Control component that exports the interface of integrated components and maps user request to an executable workflow of services. In other to flexibilize services implementation, they are implemented using the software engineering framework technique with hot-spots [15] to be instantiated. Through component reuse or adaptation of framework components the environment offers an answer for the generation of heterogeneous data integration systems tailored to specific application requirements.

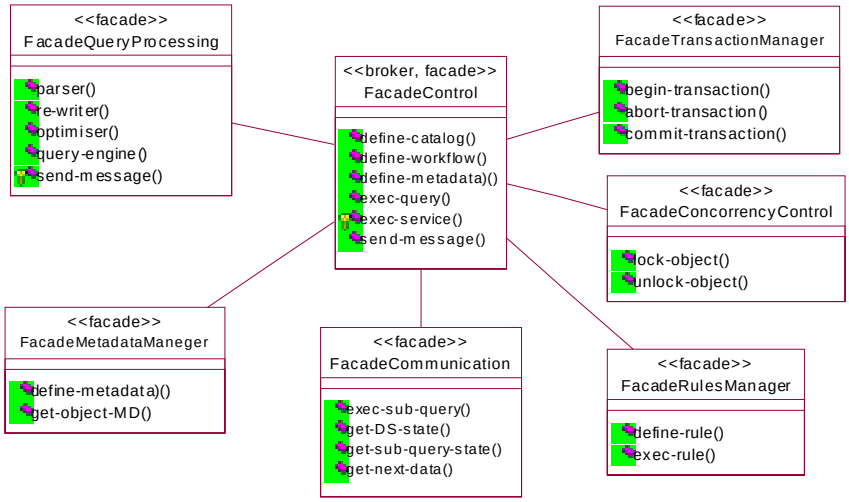

Figura 2 CoDIMS Components

A data integration middleware system is responsible for providing access to data that is distributed and stored over heterogeneous data sources. Given this general definition, the CoDIMS approach for the development of data integration systems specifies some pre-defined interfaces corresponding to data integration middleware services (DIMS) commonly presented in this type of systems, which include: Metadata Manager, Query Processing, Transaction Manager, Concurrency Control, Rule Manager and Communication (see Figure 1). For each of these interfaces we provide different components that may be selected into a configured system. In addition, the environment offers a Control component that takes part in any configuration.

New interfaces, corresponding to DIMS not initially previewed, can be added to the environment through their publication in the Control component (see section 3.2) and by providing its implementation. The added service is included in a workflow sequence to be evaluated by the Control in response to a corresponding user request. This is the strategy adopted to extend CoDIMS to support the particles tracing problem.

The flexibility obtained with these techniques can be summarized as follows:

- DIMS components: provides data integration services fucntionality.
- Framework modules: provides DIMS behavior flexibility.
- Control component: enables DIMS integration into a configured system.

DIMS component are described in the following section.

3.1. System Components Description

The CoDIMS environment includes some basic data integration functionality implemented by the so called predefined DIMS as well as some interfaces modeled to support DIMS integration and the Control module for managing such integration. In this section, we briefly describe the functionality of predefined DIMS:

- The Metadata Manager component is responsible for managing data source and global schema metadata. In a grid environment, this component transparently maps users data views to distributed data sources. User data

views are expressed according to a canonical data model adequate to the characteristics of supported applications. The set of user data views is called integrated global view. Users formulate queries over objetcs in the global view. The mapping of global views to physical data sources is also expressed through the Metadata Manager component.

- The Query Processing component (QPC) is the most important one in a CoDIMS middleware system. It provides for the efficient evaluation of queries over distributed resources within a grid environment. Within the grid, QPC is extended to support data distribution and query parallelism. It interfaces with Metadata Manager to obtain: mapping information, data sources schema, statistics and localization information.
- The Communication component is responsible for the communication between the middleware system and each data source. A wrapper translates local sub-queries into specific data source access method calls, as in others mediator systems.
- The Transaction Manager component is responsible for providing transaction ACID properties (Atomicity, Consistency, Isolation, and Durability) to applications, in cases where updates of data sources are allowed. This component will not take part into the grid middleware instance.
- The Concurrency Control component provides mechanisms for implementing the isolation of concurrent transactions. This component will not take part into the grid middleware instance.
- The Rule Manager component enriches the system with active behavior as proposed in [22]. This component will not take part into the grid middleware instance.

DIMS components are integrated via the Control component that is described in the following subsection.

3.2. The Control Component

The Control component is the essence of the CoDIMS environment. The Control stores, manages, validates, and verifies both Physical and Logical Configuration. Physical configuration corresponds to the selection of DIMS components, their customization according to application requirements, and registration in the catalog. The selection of DIMS components is subject to the set of offered and required operations that each DIMS specify. By matching the requirements of all selected DIMS in a configuration, the Control component validates it.

Logical configuration referes to the execution logic that integrates DIMS components during the evaluation of a user request. The CoDIMS approach achieves a complete adaptability in terms of services to be executed in a configured system, by associating to an user request a workflow program [25] of DIMS invocation. New configurations require the specification of a workflow program to each of its user requests.

During execution, the Control component automatically responds to client requests by scheduling DIMS services, based on its workflow, and proceeding with DIMS invocation.

Figure 3 presents the Control component class model, its facade and its three main modules. In the diagram, some specific notation expresses the facade class as extensible, symbolizing its implementation as an OO framework and static, meaning that the module is not reconfigurable during execution.

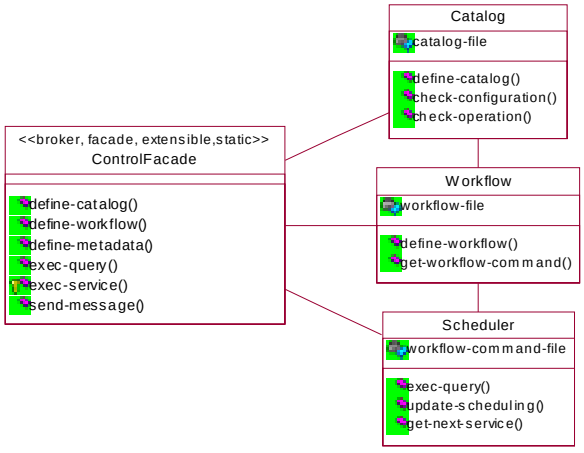

Figura 3 Control Component

The Catalog specifies the physical configuration, which registers each DIMS component present in a configuration, including its name and offered and requested operations. The Workflow module stores workflow programs associated to user requests. Finally, the Scheduler module evaluates workflow programs by following its execution logic and invoking DIMS services, accordingly.

The user of the CoDIMS environment generates a configuration through a process that we present in the next subsection.

3.3. The Configuration Process

The Control, Communication, Query Processing, and Metadata Manager components must exist in all configurations. According to the application requirements, other components may be included. The range of possible configurations of CoDIMS can vary from a midlleware with the functionality of a simple wrapper to a complex HDBMS. In cases where a full HDBMS functionality is required the middleware system could incorporate the Transaction Manager and Concurrency Control components.

The process of generating a specific configured system comprehends the following phases:

- Design: the system designer selects the necessary DIMS components. In this phase, new DIMS components may be projected or existent ones may need adaptation;
- Configuration: the system configurator registers the physical and logical configuration in the Control component. For this, two script files are executed by the operations: *define-configuration* and *define-workflow*;
- Load Metadata: the application designer (database administrator) defines local, external, and global metadata through three scripts files to be processed by the *define-metadata* function.

During the Configuration step, the Catalog module *check-configuration* method verifies if all the required services are being offered by the components interfaces that participate in the configuration. In a similar way, the *check-operation* method verifies if all operations defined in the workflow are being offered by the specific component. After all these phases, the system is configured and ready for client requests.

4. A CoDIMS Configuration for the TCP in a Grid

In this section we sketch the extension of the CoDIMS environment to support the particles trajectory computing problem (TCP) in a grid architecture.

4.1. The Particle Trajectory Computing Problem

Before discussing the middleware architecture, a more detailed presentation of the TCP problem is required. This work simulates virtual particles trajectories by interpolating their positions through a path based on velocity vectors associated to the Domain Model. Information regarding particles initial position, domain decomposition and fluid velocity vectors are given to the problem [19]. These information are modelled as relations:

- *Particles* (part-id,time-instant,point): particles in their initial position in a time instant;
- *Geometry* (id, tetrahedron): cell domain decomposition, modelled as tetrahedrons;
- *Velocity* (point,time-instant,velocity): fluid velocity in cell vertices;

In addition, we consider a program *trajectory*(particle-id, point, velocity), that computes the next position of a given particle, producing a tuple *t* (part-id,point).

The computation of particles path consists of estimating particles positions in a path for a certain time interval. Each record corresponds to an instant in time and is represented by data according to the above schema. The *TCP* aims at interpolating particles position through the fluid path between record intervals.

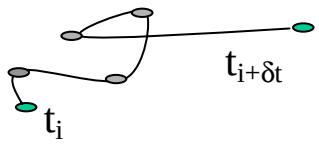

Figura 4 Interpolating particles position between instants t_i and $t_{i+\delta t}$

The computation of particles position in a time instant can be expresses as an SQL-like query embedded in a TCP procedure, such as:

Begin *TCP procedure*
for i= 1 to number-iterations do
select part-id, *trajectory(p.part-id,p.point, v.velocity)*
from *Particles p, Geometryg, Velocity v*
where
contains (p.poin,g.tetrahedron) and
matches(g.tetrahedron,v.point,p.time-instant)
end TCPprocedure

The TCP procedure computes particles subsequent positions up to the number of defined iterations. The computation is encapsulated in an *SQL-like* query. The user-function *contains* finds a tetrahedron in *Geometry* whose region contains a virtual particle, whereas *matches* captures the velocity vectors according to tetrahedron vertexes positions. The trajectory program computes the resulting velocity vector and particle´s next position in the path.

The TCP problem is an instance of a class of visualization application problems that will benefit from a middleware infra-structure capable of providing an efficient evaluation within a grid environment. Many challenges are brought by this application. Initially, we are interested in integrating the middleware within the Grid architecture and its standard software (Globus toolkit 3) platform. In respect to the TCP problem itself we will be focusing on issues related to the parallel computation of particles trajectory using available grid resources. In this initial prototype we will be concerned with three main issues: load balancing the parallel computation, managing memory resources and minimizing message exchange between nodes.

4.2. A CoDIMS middleware for Visualization applications

In order to deal with the issues raised by the TCP class of applications within a grid environment we propose a new CoDIMS middleware system configuration. The CoDIMS instance will comprehend the following components: Control, Metadata Manager, Query Processing, and Communication, see Figure 5. The Control, Metadata Manager and Communication components implement their traditional functionality, whereas the Query Processing component is adapted for supporting the TCP on a grid environment. It basically provides for parallel evaluation of particles trajectory within grid nodes.

The generated middleware completely encapsualtes the grid environment in respect to the visualization application. It provides: a set of adapted components, implemented as *web services*[27]; flexible scheduling of services coordinated by the Control component and transparent access to data sources. The main functionality is implemented in the Query Processing component that is responsible for the efficient computation of particles trajectory.

In the next section we detail the Query Processing component.

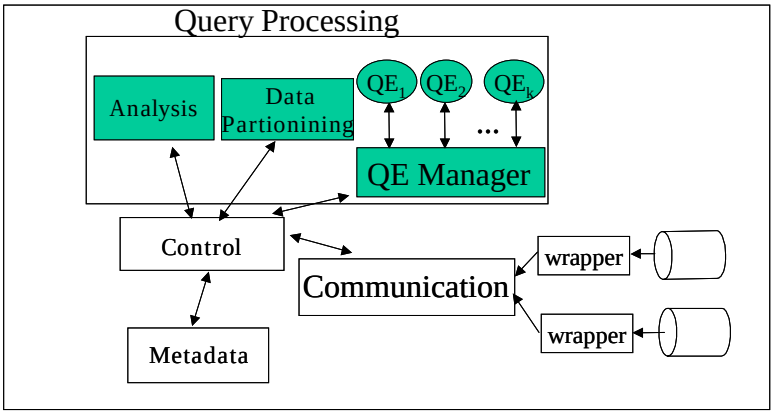

Figura 5 The TCP CoDIMS architecture

4.2.1. The Query Processing Component (QPC)

The *TC* problem once modelled as an SQL-type query and corresponding relations allows us to use query processing techniques in order to cope with issues raised in section 4.1. The grid environment can be seen as a loosely coupled query processing system for which parallel and distributed strategies have been developed. In particular, one may try to use parallel query processing techniques to deal with the issues presented in Section 4.1.

Before describing the techniques proposed to parallel the TCP, we will describe the modules designed to integrate the Query Processing component for this CoDIMS instantion.

The *Query analysis* module evaluates the syntactic and semantic correctness of user requests. *Data partitioning* receives a partitioning policy; retrieves the corresponding relation fragments and stores them in the designated node. Finally, the *Query Processing* module process a query execution plan (QEP). Figure 5 presents the configured system components: Control, Metadata Manager, Query Processing, and Communication.

```
Define Component QueryProcessing
Offered-Operations
  analyze   (int id, string procedure, int RC)
  partition  (int id, string policy, string relation, int RC)
  query-engine(int id, string op-tree, int RC )
Requested-Operations
  meta-data, get-object-MD (int id, string MD-type, string object-name, string obj, int RC)
  Communication, exec-subquery (int id, string DS-name, string subquery, string result, int RC)
  Communication, get-next-data (int id, string DS-name, string data, int RC)
End-Component

Define Workflow TCPA
  Operations
    QueryProcessing (analyze);
    QueryProcessing (partition)
    Parallel
        QueryProcessing (query-engine, query)
  End-Operations
```

Figura 6 Physical and Logical Configuration

On the top of Figure 6 we present the definition of the *Query Processing* component with its offered and requested operations, including their parameters.

In the logical configuration we provide a workflow that defines services schedule. A workflow instance is loaded by the Control component according to user requests. Next it starts its execution following the order stablished by the workflow. On the bottom of Figure 6 we present a definition for the *TCP* workflow.

Having given a general overview of the QPC component we present in more detail, in the next section, the Query Engine module.

4.2.2. The Query Engine Module

The process of computing particles trajectory is modelled through a parallel query execution plan, similar to the ones in traditional database systems [12]. A query execution plan is evaluated by the query execution module. Query evaluation will include treatment for: data partitioning, data transfer and management of local memory.

In respect to data distribution, a main concern is to efficiently partition and transfer the domain geometry dataset between grid nodes. On this matter, two main aspects are considered: the data volume initially transferred between the database and the processing nodes and the volume of message exchange during query evaluation. Secondly, regarding memory management we need also to consider out-of-core problems, i.e techniques to deal with data volumes that do not fit completely in main memory.

This first prototype solution adopts the Spatial Hash-Join(SHJ) algorithm [23] for supporting the evaluation of the *contains* user-defined function. The intuition is that we can replicate the whole domain geometry relation (Geometry) on each available node of the grid. The first step of the SHJ will spatialy split the domain geometry in *b* buckets, each measuring $size(D)/M_i$, where *D* is the size of the domain geometry representation in bytes and M_i is the size of available memory in node *i* (see Figure 7) minus a constant corresponding to the memory needed for query operators and processing tuples.

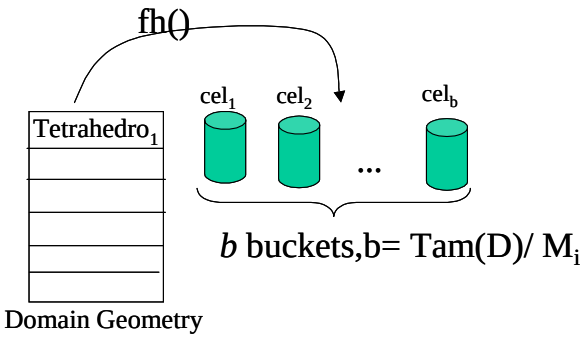

Figura 7 First step in the Spatial Hash-Join

Once the domain has been entirely processed by step1, the first bucket is copied to memory and step 2 begins. In the second step, sets of particle tuple are read and placed into an input buffer. Each particle is then processed by the *spatial hashing function* according to its current position. If the produced *bucket-id* is in memory, the tuple is directly joined with the tetrahedrons in the correspondig cell. Otherwise, it is stored in a patricle bucket for later evaluation, (see Figure 8). In Figure 8, we observe that the *trajectory* user function is treated as an operation in the query operator tree. An interesting aspect of this SHJ implementation is that a tuple produced by the join is returned to the input queue until it has passed through a specified number of iterations. This is not standard, considering traditional *hash-join* algorithm [28], and may cause extra swapping of buckets to and from memory.

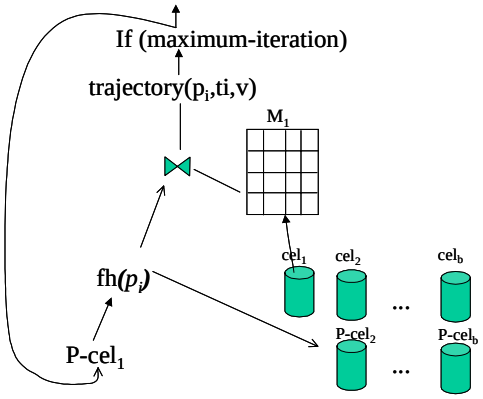

Figura 8 Spatial Hash-Join step 2

Our main motivation for adopting this strategy is to guarantee that a complete particle trajectory can be computed in a single node, eliminating inter-node communication. Furthermore, as a consequence, we can devise a partition policy regarding uniquely the number of particles to be processed by each grid node, which is the basis for a balanced execution.

A CoDIMS configuration for supporting the TCP is being implemented at the LNCC.

5. Conclusions and Future Work

The Grid environment opens new opportunities for supporting large scale applications. In this paper, we

propose CoDIMS-Grid, which is a middleware, based on grid services and distributed database technology, designed for the efficient evaluation of scientific queries over the Grid.

Our work is motivated by a particle path computation problem within the scientific visualization domain. The intuition is to build upon the experience on parallel and distributed query processing to offer efficient computation of scientific queries in the Grid environment. Particles trajectory path computation is modeled as a special case of the *bag of tasks* parallel type of problems, where for each time instant we need to compute a particle next position based only on its current position and a velocity vector in the flow. In particular, we are interested in how to parallel the computation of particles trajectory in grid nodes subject to: restrictions on the memory size of each node, in respect to the path geometry data, and the high cost of inter-node message interchange during computation.

We extended the Hash Join algorithm to support spatial hashing functions and to allow for multiple join evaluations for the same tuple, according to the number of iterations. We aim also to reduce inter-node messaging by splitting the geometry data with a spatial hashing function and locally materializing resulting buckets.

We are now implementing the first version of the CoDIMS-Grid middleware. The project opened up new problems to be investigated, such as:

- how to dynamically divide the geometry space according to the type of fluid flow;
- how to schedule grid nodes for parallel evaluation;
- how to dynamically instantiate query engine services in scheduled nodes;

References

[1] Porto, F., Giraldi,G., C. de Oliveira, J., et-all, "CoDIMS - an adaptable middleware system for scientific visualization in Grids", Concurrency and Computation: Practice and Experience, John Wiley and Sons,V.16,N.5, april, 2004.

[2] Mecca, G.; Atzeni, P.; Masci, A.; Merialdo & P. Sindoni,G. "The Araneus Web-Base Management System". *ACM SIGMOD International Conference on Management of Data*, Seattle, USA, May, 1998.

[3] Fernandez, M.; Florescu, D.; Kang, Jaewoo et al. "STRUDEL: A Web-site Management System". *ACM SIGMOD International Conference on Management of Data*, Arizona, USA, May, 1997.

[4] Molina, H. G.; Hammer, J.; Ireland, K. et al.. "Integrating and Accessing Heterogeneous Information Sources in TSIMMIS". *Journal of Intelligent Information Systems*, Vo. 8, No. 2, pp. 177-232, March, 1997.

[5] Tomasic, A.; Raschid, L. & Valduriez, P. "Scaling Access to Heterogeneous Data Source with DISCO". *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 5, pp. 808-823, September, 1998.

[6] "LE SELECT: a Middleware System for Publishing Autonomous and Heterogeneous Information Sources". INRIA, English, 1999.
(http://www-caravel.inria.fr/~xhumari/LeSelect/)

[7] Fankhauser, P.; Gardarin, G.; Lopez, M. et al. "Experiences in Federated Databases: From IRO-DB to MIRO-Web". *Proceedings of the 24th VLDB Conference*, USA, 1998.

[8] "The Garlic Project". *http://www.almaden.ibm.com/cs/garlicl* – 16/04/2000.

[9] Martinez, M.R. & Roussopoulos, N. "MOCHA: A Self-Extensible Database Middleware System for Distributed Data Sources". *ACM SIGMOD International Conference on Management of Data*, Dallas, USA, 2000.

[10] Foster, I., Kesselman, C., Nick, J.M. and Tuecke, S.,"The Physiology of the Grid: An Open Grid Services Architecture for Distributed System integration", www.globus.org/research/papers/ogsa.pdf , 2002

[11] Bernstein, P., Brodie, M., Ceri, S. et al. "The Asilomar Report on Database Research". *SIGMOD Record*, Vol. 27, No. 4, December, 1998.

[12] Silberschatz, A. & Zdonic, S. "Database Systems – Breaking Out the Box". *SIGMOD Record, Vol. 26*, No. 3, September 1997.

[13] Carey, M.J., DeWitt, D.J., Graefe, G., Haight, D.M., Richardson, J.E., Schuh, D.T., Shekita, E.J., and Vandenberg, S.L. "The EXODUS Extensible DBMS Project: an Overview", in Maier, D., and Zdonik, S. (editors), *Readings on Object-Oriented Database Systems*, Morgan-Kaufmann, 1990.

[14] Batory, D.S., Barnett, J.R., Garza, J.F., Smith, K.P., Tsukuda, K., Twichell, B.C., and Wise, T.E. "GENESIS: An Extensible Database Management System", in Maier, D., and Zdonik, S. (editors), *Readings on Object-Oriented* Database Systems, Morgan-Kaufmann, 1990.

[15] Nierstrasz, O. & Dami, L. "Component-Oriented Software Technology", *In Object-Oriented Software Composition*, Chapter 1, edited by Nierstrasz, O & Tsichritzis, D. - Prentice-Hall Inc., 1995.

[16] Pree, W. "Component-Based Software Development - A New Paradigm in Software Engineering? ". *Software-Concepts and Tools* (18), Springer-Verlag, 1997.

[17] Rosenblum, L. et all., "Scientific Visualization: Advances and Challenges", Academic Press,1994.

[18] Lane, A.L**,** "Parallelizing a Particle Tracer for Flow Visualization", http://www.nas.nasa.gov.

[19] Hamann, B., Wu, D and Moorhead II, R. J., "On Particle Path Generation Based on Quadrilinear Interpolation and Bernstein-Bézier Polynomials", on IEEE Trans. on Visualization and Comp. Graph, v.1, n. 3, september 1995.

[20] Stolk, J. and van Wijk, J.J., "Surface-Particles for 3D Flow Visualization", on Advances in Scientific Visualization, Springer-Verlag, Berlin, Heidelberg, 1992, pp.119-130.

[21] Barnard,S.,Biswas,R., Saini, S., Van der Wijngaart, R., Yarrow, M. and Zechtzer, L., "Large-Scale Distributed Computational Fluid Dynamics on the Information Power Grid using Globus", on The 7th Symposium on the Frontiers of Massively Parallel Computation, 1999.

[22] Widom, J., and Ceri, S., editors. Active Database Systems - Triggers and Rules For Advanced Database Processing. Morgan-Kaufmann, 1996.

[23] Lo, M.L., Ravishankar, V.,"Spatial Hash-Joins", ACM-SIGMOD Intl. Conf. On Management of Data, Montreal, Canada, 1996.

[24] Gamma, E.; Helm, R.; Johnson, R. & Vlissides, J. "Design Patterns – Elements of Reusable Object-Oriented Software ". *Addison-Wesley professional computing series*, 1995.

[25] Jablonski, S. & Bussler, C. "Workflow Management – Modeling Concepts, Architecture, and Implementation". International Thomson Computer Press, 1996.

[26] Barbosa, A.C.P.. "Middleware Para Integração de Dados Heterogêneos Baseado em Composição de Frameworks". PhD theses, PUC-Rio, Brazil, may 2001 (In Portuguese).

[27] **http://www.w3.org/2002/ws/**

[28] Garcia-Molina,H.,Ullman, J.,Widom,J., "Database System Implementation ", Prentice-Hall, 2000.