CoDIMS: an adaptable middleware system for scientific visualization in Grids

F. Porto¹, G. Giraldi²,*,†, J. C. Oliveira², R. Silva² and B. Schulze²

¹Department of Systems Engineering, Military Institute of Engineering, Rio de Janeiro, RJ, Brazil
²National Laboratory for Scientific Computing, Department of Computer Science, Av. Getulio Vargas 333, 25651-075, Petrópolis, RJ, Brazil

SUMMARY

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 environment. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: Grid computing; middleware; scientific visualization; particle tracing

1. INTRODUCTION

CoDIMS (configured data integration middleware system) [1] is a middleware environment for the generation of adaptable and configurable middleware systems. Data integration systems [2–4] 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.

*Correspondence to: Gilson Giraldi, National Laboratory for Scientific Computing, Department of Computer Science, Av. Getulio Vargas 333, 25651-075, Petrópolis, RJ, Brazil.
†E-mail: gilson@lncc.br

Contract/grant sponsor: PCI/LNCC Program
Contract/grant sponsor: CNPq; contract/grant number: 300840/1999-3

Copyright © 2004 John Wiley & Sons, Ltd.
With CoDIMS we aim to provide an architecture that can be adapted to new data integration application requirements, so that lightweight 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 generate middleware systems to support the execution of scientific applications in a Grid environment. As an example of such applications, we consider the computation of particles’ (i.e. virtual particles) trajectories for a flow visualization application. The trajectory computing problem (TCP) combines the processing of huge datasets, storing particle 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 experimentation with different strategies for the efficient computation of particles’ trajectories in a flow through the use of a Grid 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 present some background on data integration and on the fluid dynamics problem. Next, in Section 3 we discuss the CoDIMS approach and its architecture. Section 4 presents the particles, tracing problem and specifies the middleware components necessary to support the application in a Grid environment. Finally, Section 5 concludes and points to future work.

2. BACKGROUND AND RELATED WORK

There are quite a few data integration middleware systems in the literature, such as MOCHA [4] and LeSelect [3]. In addition to supporting data integration, these systems provide mechanisms for user defined function execution. In this scenario, user queries can be seen as simplified workflows over distributed and heterogeneous data and programs.

An important aspect of these systems is that, in general, they are designed to have a specific application in mind. Supporting new application requirements may entail quite a strong developing effort. Conversely, systems designed for supporting a wide spectrum of applications are commonly large and heavy, which translates into execution inefficiency and complexity in use.

The middleware proposed aims at addressing scientific visualization problems. As an example of such applications we will consider particle tracing problems [5–8]. They can be mathematically defined by an initial value problem [6,9]:

\[ \frac{dx}{dt} = F(x, t), \quad x(0) = P_0 \]  

(1)

where \( F : \mathbb{R}^3 \times \mathbb{R}_+ \to \mathbb{R} \) 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 [6].

The problem (1) in general does not have an analytical solution. Thus numerical methods must be used [9,10]. These methods basically generate the particle trajectory step by step, following some scheme for field interpolation inside cells [7].

3. THE CoDIM S APPROACH OVERVIEW

In this section, we present the CoDIM S approach overview. CoDIM S is a flexible and configurable environment to generate data integration middleware systems. The configuration is obtained through a Control component that exports the interface of integrated components.

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

New interfaces, corresponding to DIM S not initially previewed, can be added to the environment through their publication in the Control component (see Section 3.1) and by providing its implementation. This will be the strategy in extending CoDIM S to support the particles tracing application.

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

- DIMS components: allow for the publication of DIMS.
- Framework modules: provide DIMS behavior flexibility.
- Control component: enables DIMS integration into a configured system.

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

3.1. The Control component

The Control component is the essence of the CoDIM S 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 restrictions. The restrictions are specified as a set of offered and required operations. By matching restrictions between selected DIMS components in a configuration, the Control component validates it. The idea behind logical configuration is to extract from the system the integration logic modeled by components interaction. The CoDIM S approach achieves a complete adaptability in terms of services to be executed in a configured system, modeling their interaction through a workflow of DIMS invocation. A predefined workflow covers basic data integration operations; new functionality, however, may be added to the system requiring the definition of a specific workflow to be evaluated in the event of the corresponding command request. The adoption of a workflow model brings flexibility in adapting to new query languages, besides detaching DIMS components. During execution, the Control component automatically responds to client requests by scheduling DIMS services, based on its workflow, and proceeding with DIMS invocation. The communication between the Control and DIMS components are carried on using RMI calls to their public interface.
The Control component comprehends three main modules: a Catalog, a Workflow and a Scheduler. The Catalog specifies the physical configuration, which registers each DIMS component present in a configuration, including its name and the offered and requested operations. The Workflow module is responsible for the logical configuration, which registers the mapping between DIMS components operations for each global command request. The Scheduler module consults the workflow in order to schedule the respective operations in execution time.

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

3.2. 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 middleware with the functionality of a simple wrapper to a complex heterogeneous DBMS (HDBMS). In addition, given the flexibility obtained by the configuration process, new functionality may be introduced leading to especially tailored data integration middleware systems.

The generation of a specific configured system incorporates the following phases.

- Design: the system designer selects the necessary DIMS components. In this phase, new DIMS components may be projected or existing 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 the exported, global and external views of integrated data through three script files to be processed by the define-metadata operation.

During the configuration step, the check-configuration method of the Catalog module in the Control component verifies if all the required services are being offered by the component 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 particle trajectory computing problem (TCP) in a Grid environment.

4.1. The CoDIMS TCP

The problem is to devise the services needed for the path of computing particles in a Grid architecture. The input datasets include information on fluid particles’ initial position, the geometry of the cell
domain decomposition and fluid velocity vectors associated with the cell’s vertices [7]. These datasets are modeled as the following relations.

1. **Particles** (part-id, time-instant, point): particles in their initial position in an instant.
2. **Geometry** (id, point): cell domain decomposition.
3. **Velocity** (point, time-instant, intensity): the velocity of fluid 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 (particle-id, point).

The computation of particle paths consists of registering particles’ positions in a certain frequency. Each record represents an instant in time and generates data according to the above schema. The TCP aims at interpolating particles’ velocity through a fluid path between record intervals (Figure 1).

The computation of particles’ positions in a time instant can be expressed as a SQL-like query embedded in a TCP procedure, such as

```
Begin TCP procedure
for i = 1 to number-iterations do
    Select trajectory(p.part-id, p.point, v.velocity)
    From Particles p, Geometry, Velocity v
    Where p.point matches i.point and i.point = v.point and p.time-instant = ‘ti’
endfor
end TCP procedure
```

The TCP procedure computes particles’ subsequent positions up to the number of defined iterations. The computation is encapsulated in a SQL-like statement, where the expensive predicate *matches* finds the cell in the domain decomposition that contains the particle’s current position. The trajectory program is modeled as a user-defined function receiving tuples resulting from spatial joining the relations *Particles, Domain Decomposition* and *Velocity*.

The TCP problem is an instance of a class of visualization application problems that will benefit from a middleware infrastructure capable of providing an efficient evaluation within a Grid environment.
In particular, we are interested in the problem of load balancing the executions between Grid nodes, the restrictions on available main memory on each node as well as minimizing message exchange between nodes.

4.2. A CoDIMS middleware for visualization applications

The configured middleware system includes the following components: Control, Metadata Manager, Query Processing, and Communication, see Figure 1. The Control, Metadata Manager and Communication components implement their traditional functionality, whereas the Query Processing component (QPC) is adapted for supporting the visualization application in a Grid environment. The generated middleware completely encapsulates the Grid environment, providing a set of adapted components and flexible scheduling of services.

In the next subsection we comment on the QPC.

4.2.1. The Query Processing component

On considering the TC problem, the QPC must be adapted to support a partitioning policy that would keep the use of Grid resources balanced while maintaining a minimum number of message exchanges between Grid nodes, during the trajectory computation. However, we need to devise adequate algorithms to deal with the out-of-core problem; that is, techniques to deal with data volumes that do not fit completely in the main memory.

Thus, a Query Processing DIMS component is designed for paralleling the execution of the TCP procedure. It is split into three modules: Query analysis, Data Partitioning and Query Engine.

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 Grid nodes. Finally, the Query Engine processes a query execution plan. As illustrated in Figure 1(a), the Query Engine component is replicated through the Grid nodes where the parallel computation would take place.

At the top of Figure 2, we present a definition of the QPC interface with its offered and requested operations. The former exports the services supported by the QPC component, while the latter is used by the Control component during the physical configuration validation.

At the bottom of Figure 2, we illustrate a logical configuration through the definition of a workflow for scheduling the QPC execution. Note that a parallel primitive was introduced to inform Control about the service to be executed in parallel.

4.3. The Query Engine module

The Query Engine module is an extended relational query engine in the sense that it evaluates traditional and user-defined operations, partially ordered by a query execution plan. An example of a user-defined operation is the trajectory program.

Each Grid node that participates in the query evaluation will receive a Query Engine instance that will be responsible for processing the data partition allocated to that node.

Our first experiment with the TCP will consider the adaptation of the Spatial Hash-Join [11] algorithm for combining particles with the corresponding velocity vector. The aim is to split the...
SUPPORTING DISTRIBUTED VISUALIZATION IN GRIDS 521

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 TCP
Operations
   QueryProcessing (analyze);
   QueryProcessing (partition)
   Parallel
      QueryProcessing (query engine);
End-Operations

Figure 2. QPC component configuration.

Geometry and Velocity relations into \( k \) partitions of size \( m/n \), where \( m \) corresponds to the number of points in the domain decomposition and \( n \) corresponds to the number of pairs of tuples (point, velocity) that fit in the main memory. By doing this, we guarantee that a complete particle trajectory can be computed in a single node, eliminating inter-site communication. Furthermore, as a consequence, we can devise a partition policy regarding only the number of particles to be processed on each Grid node, which is the basis for a balanced execution.

Thus, the query execution plan, produced by the Query analysis component from the TCP procedure, will consist of a spatial join and the invocation of the trajectory program on the tuples resulting from the join. The middleware for supporting the TCP is to be implemented at the LNCC.

5. CONCLUSIONS AND FUTURE WORK

Nowadays, there is an increasing demand for accessing and integrating heterogeneous and distributed information, such as those available in a Grid environment. Designing flexible middleware systems to process and integrate data from a great variety of applications served by a Grid environment is a great challenge. This work proposes the use of CoDIMS, a flexible environment for the configuration of data integration middleware systems, to support the generation of adaptable middleware systems for the Grid environment. The CoDIMS provides software adaptability through the combination of three techniques: first, the selection and integration of adequate components through the physical configuration; second, the customization of each component through the instantiation of an object-oriented framework; and third, the logical configuration that allows for the scheduling of services according to requests of client applications. We show how CoDIMS can be adapted to generate a middleware system for supporting the execution of a scientific visualization application in a...
Grid environment. We describe the middleware components needed for efficiently computing particle paths. We design a query processing strategy that keeps to a minimum the communication between Grid nodes while providing the basis for a balanced execution. We now intend to implement the first prototype of the middleware system and use it for the validation of the TCP problem, with different parallel strategies, in Grids.

REFERENCES

2. Tomasic A, Raschid L, Valduriez P. Scaling access to heterogeneous data source with disco. *IEEE Transactions on Knowledge and Data Engineering* 1998; 10(5):808–823.