Parallelism in Processing Queries on Complex Objects

T. Härder **H. Schöning** **A. Sikeler**

University Kaiserslautern, Department of Computer Science,
P.O. Box 3049, D-6750 Kaiserslautern, West Germany

Abstract

lex objects to support non-standard database applications re the use of substantial computing resources because their rful operations and their related integrity constraints must rformed and maintained in an interactive environment. Since xploitation of parallelism within such operations seems to be ising, we investigate the principal approaches for processing ery on complex objects (molecules) in parallel. A number of ments favor methods based on inter-molecule parallelism as st intra-molecule parallelism. Retrieval of molecules may be ized by multiple storage structures and access paths. Hence, tenance of such storage redundancy seems to be another good cation area to explore the use of parallelism. Deferred update s to be a bad idea, whereas concurrent update strategies porate salient application features. For performance reasons, ave chosen a multiprocessor system sharing an instruction essable common memory which is used for buffer gement, synchronization, and logging/ recovery. Activation of rrent requests is supported by a nested transaction concept allows a safe and effective execution control within parallel ns of an operation.

1. Introduction

standard database applications such as 3D-modeling for pieces or VLSI chip design [1] require adequate modeling ties for their application objects for various reasons. If the t representation is adjusted to the needs of a particular cation area, the intended object handling (e.g. composition or formation of objects) may be performed in a natural and th way, thereby saving a lot of ponderous deviation steps. models supporting such applications embody some degree of t orientation (towards the application objects). The notion of lex objects is used to indicate that such objects have an nal structure (structured components) maintained by the base management system (DBMS) and that access is provided e object as a whole as well as to its components (structural t orientation). To enhance integrity control and semantic ssiveness, more object properties beyond the structural onships have to be specified and preserved by the data model vioural object orientation). Such a rich data model supports priate forms of data abstraction and encapsulation (e.g. s) which relieve the application from the burden of taining intricate object representations and checking lex integrity constraints.

he other hand, DBMS requests using such a high-level object-ted DBMS interface (incorporated by a set of powerful ADTs) very long execution path lengths since all aspects of complex object handling have to be performed inside the DBMS. Wh applying processing concepts known from conventional DBM serious performance problems may occur in terms of respon time, e.g. in an interactive construction environment. Although operation is decomposed into a tree of suboperations (Fig. 1b), t classical DBMS processing strategy observes a synchrono activation of each suboperation and its strictly serial executi Only some conventional systems deviate marginally from t processing strategy by using low-level parallelism for certa house-keeping tasks, e.g. writing modified pages to disk. general, however, concurrent execution on behalf of a u operation is not achieved [2].

The use of intra-transaction parallelism for higher-le operations was investigated in a number of database machi projects [3]. These approaches focus on the exploitation parallelism in the framework of the relational model. Comp relational queries are transformed into an operator tree relational operations in which subtrees are executed concurren (evaluation of subqueries on different relations) [4]. Oth approaches utilize special storage allocation schemes distributing relations across multiple disks. Parallelism achieved by evaluating the same subquery on the vario partitions of a relation [5, 6].

We are going to investigate possible strategies to expl parallelism when processing complex objects. In order to specific, we have to identify our concepts and solutions in t framework of a particular data model and a system desi facilitating the use of parallelism. Therefore, we refer to t molecule-atom data model (MAD model [7]) which is implement by an NDBS kernel system called PRIMA [8]. We use the te NDBS to describe a database management system tailored to t support of non-standard applications.

For this purpose, we introduce the essential concepts of an ND architecture and a model of NDBS operations. We focus on t principal ways to process a query on complex objects in paral Furthermore, we consider the use of parallelism when redunda storage structures kept for performance reasons have to maintained. In order to achieve a safe and effective executi control for parallel actions, we tailor the concept of nest transactions to our distributed processing strategies in a serv complex carrying the PRIMA code. Finally, we conclude with summary of our design proposals.

2. A Model of NDBS Operations

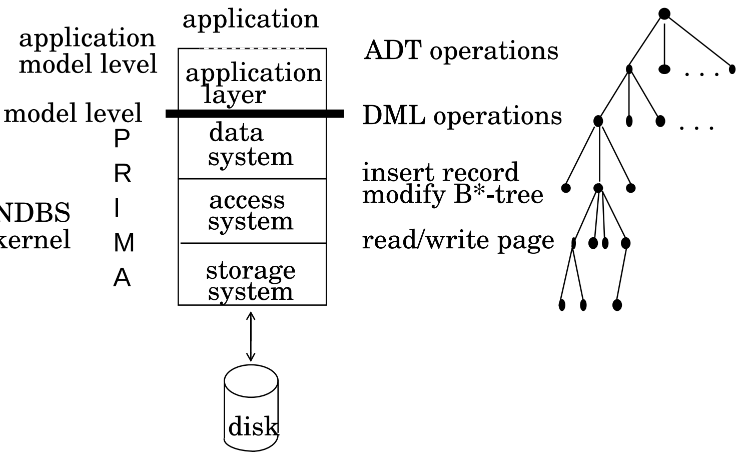
In order to describe our concepts of supporting parallelism in t framework of NDBS processing, we introduce a multi-laye r concurrent processing. On the other hand, our architecture own in Fig. 1a reflects a high degree of data independence tained by explicit interfaces among layers.

verall architecture consists of a so-called NDBS kernel and a ber of different application layers, which map particular cations to the data model interface of the kernel. Our kernel h is considered application-independent is divided into three s:

e storage system provides a powerful interface between main emory and disk. It maintains a database buffer and enables cess to sets of pages organized in segments [8].

e access system manages storage structures for basic objects lled atoms and their related access paths. For performance asons, multiple access paths and redundant storage structures may be defined for atoms.

e data system dynamically builds the objects available at the ta model interface. In our case, the kernel interface is characterized by the MAD model and its language called MQL (molee query language) [7]. Hence, the data system performs composition and decomposition of complex (structured) objects lled molecules.



architecture of an NDBS

b) control flow for a user operation

Figure 1: Model of operation

pplication layer uses the complex objects and tailors them to more complex) objects according to the application model of en application. This mapping is specific for each application (e.g. 3D-CAD, VLSI design, geographic information gement). Hence, for each application area a different cation layer exists which provides a tailored interface (e.g. in of a set of ADT operations) for the corresponding application.

NDBS architecture as exhibited in Fig. 1a lends itself to a station-server environment in a smooth and natural way. The cation and the corresponding application layer are dedicated vorkstation, whereas the NDBS kernel is assigned either to a e server processor or to a server complex consisting of ple processors. This architectural subdivision is strongly tated by the properties of the MAD model: Sets of molecules h consist of sets of heterogeneous atoms may be specified as ssing units.

far, we have sketched the static mapping of NDBS objects operations organized in hierarchical layers. The dynamic ssing of user operations may be explained by a tree of perations as illustrated in Fig. 1b. This operation tree reflects elementary operations: "Each call to a subroutine is an exampl a primitive at one level of implementation invoking a set primitives at a lower level of control [9]".

In conventional DBMS, all operations in such an operation tree called synchronously and are executed serially (left-most dep first traversal). Given an appropriate run-time environme operations at a certain level may be called in parallel, i.e. corresponding subtrees as shown in Fig. 1b are execu (traversed) concurrently. In principle, such a decomposition a parallel execution is conceivable at every level of operati However, certain prerequisites such as sufficient operat granules are necessary for successful application of concurrency processing a user operation. In this paper, we want to focus concepts to exploit parallelism within the NDBS kernel, that how concurrent and asynchronous actions should be organized the server complex carrying the NDBS kernel code.

2.1 The Data System Interface

In order to describe the concepts for achieving parallelism sufficient detail, we have to refine our view of the ker architecture and the interfaces involved. It is obvious that the d model plays the major role and determines many essential fact which enable reasonable parallelism: sufficiently large d granules, set orientation of request, dynamic construction objects (result sets), flexible selection of processing sequences,

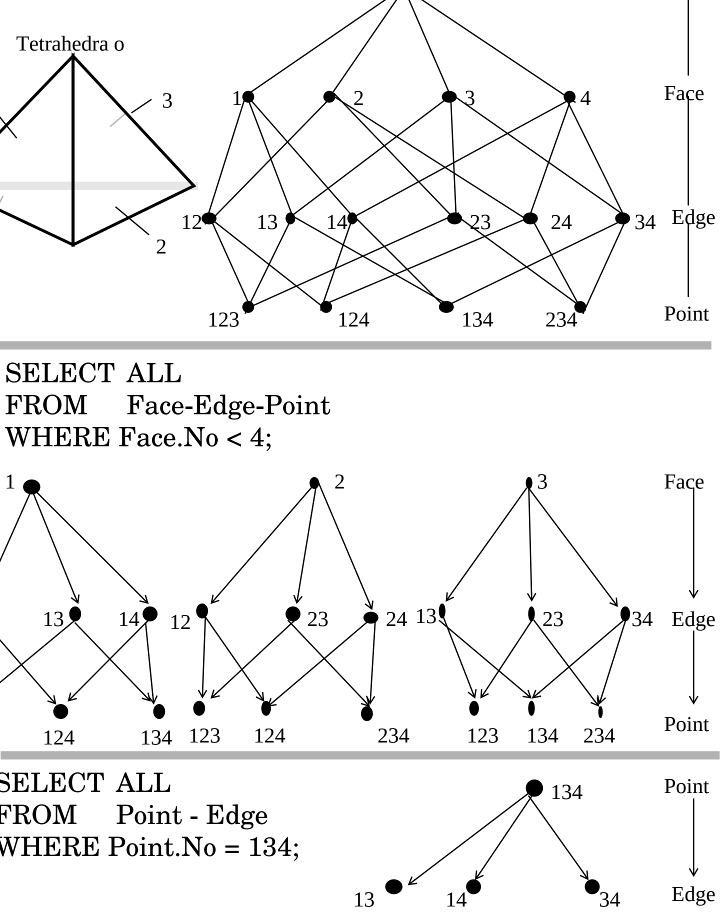
In our system, the data model interface is embodied by the M model and its language MQL which is similar to the well-kno SQL language. Here, we cannot introduce this model with all complex details, but only illustrate the most important conce necessary for our discussion. In the MAD model, atoms are used a kind of basic element (or building block) in order to repres entities of the real world. In a similar way to tuples in relational model, they consist of an arbitrary number of attribu The attributes' data types can, however, be chosen from a ric selection than in the relational model, i.e. apart from conventional types the type concept includes

- the structured types RECORD and ARRAY,
- the repeating group types SET and LIST, both yielding a po erful structuring capability at the attribute level as well as
- the special types IDENTIFIER (serving as surrogates) for id tification purposes and REF_TO for the connection of atoms

Atoms are grouped to atom types. Relationships between ato are expressed by so-called links and are defined as link ty between atom types. Link types are treated in a symmetric w i.e. links may be used in either direction in the same manner. S link types directly map all types of relationships (1:1, 1:n, n: The flexibility of the data model is greatly increased by this dir and symmetric mapping. Link types are represented by a pai REF_TO attributes (reference and "back-reference") one in eit involved atom type. For example, a link type may be specified follows:

- FIDs: SET_OF (REF_TO(Face.EIDs)) in an atom type *Edge*
- EIDs: SET_OF (REF_TO(Edge.FIDs)) in an atom type *Face*.

In the database, all atoms connected by links form mesh structures (atom networks) as illustrated in Fig. 2.:



```
SELECT ALL
FROM    Face-Edge-Point
WHERE Face.No < 4;
```

```
SELECT ALL
FROM    Point - Edge
WHERE Point.No = 134;
```

Figure 2: Dynamic construction of molecules
from a sample geometric object

cules are defined dynamically by using MQL statements and
to be derived at run-time. Each molecule belongs to a
cule type (which is specified in the FROM clause). The type
iption establishes a connected, directed and acyclic type
(cycles occur when recursive types are specified), in that a
ng point (i.e. root atom type) and all participating atom and
types (for short "-") are specified. A particular example of a
cule type is Face-Edge-Point. Such a molecule type
mines both the molecule structure as well as the molecule set
groups all molecules with the same structure. At the
ptual level, the dynamic construction of molecules proceeds
traight-forward manner using the molecule type description
template: For each atom belonging to the root atom type all
ren, grandchildren and so on are connected to the molecule
ture, terminating after all leaves of the molecule structure
eached. Connecting children to the molecule structure means
rming the hierarchical join which is supported by the link
pt. Hence, for each root atom a single molecule is constructed.
b shows the result of a molecule construction for Face-Edge-
molecules, where the set of molecules was restricted.
hermore, it illustrates the most important properties of the
interface

MQL request handles a set of molecules.

e molecules as complex objects consist of sets of atoms of dif-
ent type, i.e., they are embodied by sets of interrelated heter-
eneous record structures.

olecule construction is dynamic and allows symmetric use of
e atom networks (e.g. *Point-Edge* (Fig. 2c)).

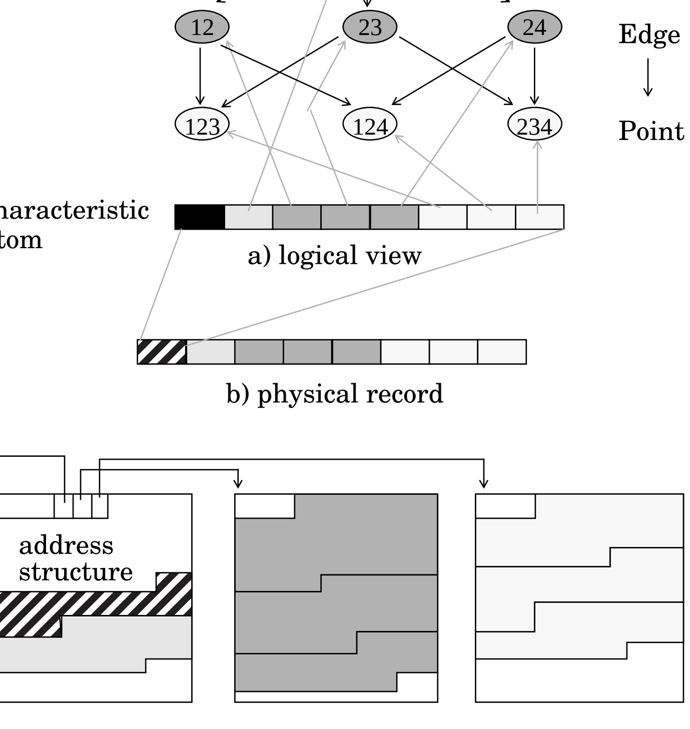
The access system provides an atom-oriented interface w
allows for direct and navigational retrieval as well as for
manipulation of atoms. To satisfy the retrieval requirements of
data system, it supports direct access to a single atom as well
atom-by-atom access to either homogeneous or heterogene
atom sets.

Manipulation operations (insert, modify, and delete) and dir
access operate on single atoms identified by their logical addr
(or surrogate) which is used to implement the IDENTIFI
attribute as well as the REF_TO attributes. Perform
manipulation operations, the access system is responsible for
automatic maintenance of the referential integrity defined by
REF_TO attributes. Thus, a manipulation operation on such
attribute requires implicit manipulation operations on ot
atoms in order to adjust the appropriate back references. Th
operations however, are triggered by a special consiste
manager (section 3.3).

Different kinds of scan operations are introduced as a concept
manage a dynamically defined set of atoms, to hold a curr
position in such a set, and to successively deliver single ato
Some scan operations, however, are added in order to optim
retrieval access. Therefore, they may depend on the existence o
certain storage structure (defined by the database administrat

- The atom-type scan delivers all atoms in a system-defined or
utilizing the basic storage structure which exists for each at
type.
- The access-path scan provides appropriate means for fast val
dependent access based on different access path structures su
as B-trees, grid files, and R-trees.
- The sort scan processes all atoms following a specified sort
terion also utilizing the basic storage structure of an atom ty
However, sorting an entire atom type is expensive and time c
suming. Therefore, a sort scan may be supported by an additi
al storage structure, namely the sort order.
- The atom-cluster scan speeds up the construction of frequen
used molecules by allocating all atoms of a corresponding mo
cule in physical contiguity using a tailored storage structure
a so-called atom cluster. For example, in Fig. 2 each *Face* at
and all its associated *Edge* and *Point* atoms may be organized
form an atom cluster (Fig. 3). On a logical level, an atom clus
corresponds to a molecule. It is described by a special so-cal
characteristic atom which consists of references to all atoms
longing to the molecule. This characteristic atom together w
all the referenced atoms is mapped onto a single physical rec
which in turn is stored in a set of pages.

The underlying concept is to make storage redundancy availa
outside the access system by offering appropriate retrie
operations (i.e. the choice of several different scans for a particu
access decision by the optimizer of the data system), whereas
the case of update operations storage redundancy has to be hidd
by the access system. As a consequence, maintaining stora
redundancy in an efficient way is a major task of the access syste
However, sequential update of all storage structures existing fo
corresponding atom results in a lack of efficiency which is
acceptable. Therefore, exploiting parallelism seems to be a natu
way to speed up a single manipulation operation.



a) logical view

b) physical record

c) mapping onto a set of pages

Figure 3: Example of an atom cluster

2.3 Activation of Concurrent Requests

e we start to evaluate our concepts for achieving parallelism rform data system and access system functions, we briefly h the process (run-time) environment of our DBMS kernel IA. In order to provide suitable computing resources, PRIMA apped to a multi-processor system, i.e., the kernel code is

synchronization of concurrent accesses. Due to the frequency references (issued from concurrent tasks) accessibility of data a synchronization of access must be solved efficiently.

In a so-called DB-distribution architecture [10] every (loos coupled) processor handles a partition of the database; suc static data subdivision and allocation is obviously in conflict w our processing requirements. Therefore, DB-sharing architectu [11] where multiple DBMS share the database at the disk le seem to be much more flexible and appropriate for our purpose a loosely coupled system, however, each DBMS has its own syst buffer creating the need to cope with fully replicated data; modification of a page in a buffer makes all copies of this page other buffers obsolete (buffer invalidation problem). Furthermo the lack of common memory enforces message-ba communication for inter-process cooperation (e.g. concurre control) which seems to be by far too slow for our type application. As a consequence, use of shared memory for crit functions is mandatory for performance reasons. Hence, we ha designed our system to run on a server complex where instructi addressable common memory [12] is available for bu management, synchronization, and logging/recovery. Such architecture (sometimes called closely coupled) avoids bu invalidation (only one buffer) and provides memory-based mess exchange as well as instruction-based synchronization primiti for shared data accesses (e.g. a "compare and swap" instructio Fig. 4 gives a short illustration of the major architectural issu Our experimental system for the server complex consists of up five processors with sufficient private memory, the comm memory, an attachment to a file processor, and a high-bandwi communication system for coupling the server processors workstations carrying the application layers and corresponding applications..

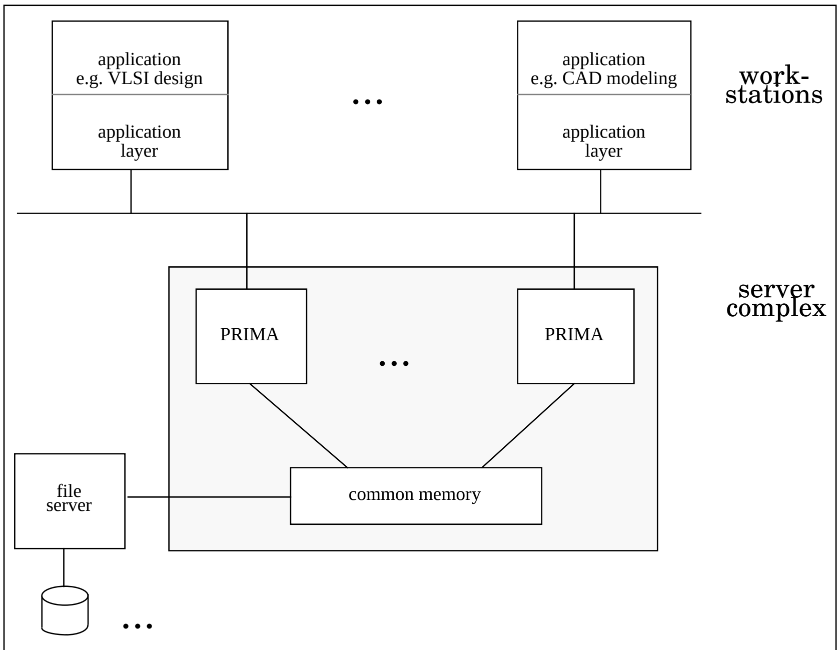


Figure 4: Run-time environment of PRIMA

instance of PRIMA (running on a particular processor) is vided into a number of processes. Each process may initiate rbitrary number of tasks which serve as run-units for the tion of single requests. Cooperation among processes is rmed by establishing some kind of client-server relationship; the calling task in the client process issues a request to the ser process where a task acts upon the request and returns an ansv to the caller. In our model, a client invokes a ser asynchronously, i.e. it can proceed after the invocation, and her