Parallel Database Management System: Kappa-P

ABSTRACT

Kappa-P (Knowledge Application-Oriented Advanced Database Management System) is a parallel database management system running on PIM machines. Kappa-P provides efficient database management facilities for knowledge information processing systems and knowledge base management systems.

KEY FEATURES

Nested Relational Model

In order to treat complex structured data efficiently, a nested relational model is adopted. With its set constructor and hierarchical attributes, we can represent complex data naturally, and can avoid the unnecessary division of relations.

Parallel Processing Depending on Data Placement

As Kappa-P has a query processing capability at each cluster, the distribution of relations and the horizontal partition of relations give us inter-cluster parallelism. Replication of a relation decentralizes access to the relation. A *quasi main memory relation* is provided as a replicated relation by using intra-cluster parallelism.

Interaction between Kappa-P and Applications

Kappa-P provides functions which reduce communication overhead between Kappa-P and various applications. Parts of the applications are executed at clusters at which a query is processed.

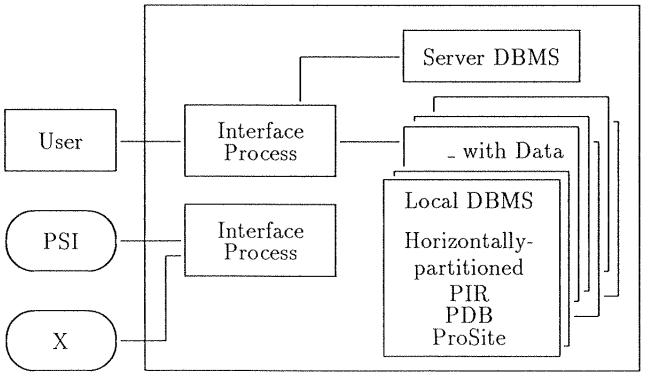


Figure: Configuration of Kappa-P

Objective of Kappa-P

The objective of Kappa-P is to provide database management facilities for many KIPSs, for instance natural language processing systems with electronic dictionaries, proof checking systems with mathematical knowledge, and genetic information processing systems with molecular biological data. Kappa-P has been developed to manage large amount of complex structured data efficiently.

Nested Relational Model

In order to treat complex structured data efficiently, the conventional relational model must be extended. In Kappa-P, a nested relational model with a set constructor and hierarchical attributes can represent complex data naturally, and can avoid the unnecessary division of relations. Moreover, the semantics of the model matches the knowledge representation language *Quixote*, which is the upper layer of the KBMS of the FGCS project. Kappa-P has charge of the database engine of this system.

Term is added as a data type in order to store various types of knowledge. The character code of the PIM machine is based on 2-byte code, but the code wastes secondary memory space. In order to store a huge amount of data, data compression and index facilities have been improved.

Configuration

The configuration of Kappa-P corresponds to the architecture of the PIM machine, and distinguishes inter-cluster parallelism from intra-cluster parallelism. Kappa-P consists of a collection of *element DBMS*s located in clusters. These element DBMSs cooperate in processing a query.

The global map of relations is managed by element DBMSs called *server DBMS*s. Server DBMSs manage not only the global map but also ordinary relations. Element DBMSs, except server DBMSs, are called *local DBMS*s. *Interface processe*s are created to mediate between application programs and Kappa-P, and to receive and send messages such as queries and answers.

Data Placement

The placement of relations also corresponds to parallelism: inter-element DBMS placement and intra-element DBMS placement.

In order to use inter-cluster parallelism, relations can be located in several element DBMSs. A simple case is the distribution of relations like

distributed DBMSs. When a relation needs a lot of processing power and a higher disk access bandwidth , the relation can be declustered as a horizontally partitioned relation and located in element DBMSs. When a relation is frequently accessed, some replicas of the relation can be made and located in element DBMSs. However, in the current implementation, the replicated relation can be used only for the global map, that is, for server DBMSs.

Relations can be located in main memory and/or secondary memory in an element DBMS. Relations which are located only in main memory are temporary relations. *Quasi main memory relations* both in secondary memory and in main memory provide guarantees that the modifications are reflected in the secondary memory.

Query Processing

There are two types of command for query processing: *primitive commands* and *KQL*, a query language based on extended relational algebra. Primitive commands are the lowest operations for relations, and can treat relations efficiently. KQL is syntactically like KL1. New operations can be defined temporarily in a query.

A query in KQL is translated into sub-queries in intermediate operations for extended relational algebra, and is submitted to the relevant element DBMSs. A query in primitive commands is submitted to the relevant element DBMSs. The query is processed as a distributed transaction among the relevant element DBMSs, and is finished under the control of a two-phase commitment protocol.

Parallel Processing

Kappa-P parallel processing takes the form of inter-cluster parallelism among element DBMSs and intra-cluster parallelism in an element DBMS. Inter-cluster parallelism provides more processing power, but also increases communication overhead. The trade-off is between processing power and communication overhead.

Intra-cluster parallelism is suitable for a DBMS manipulating large amounts of data. Kappa-P uses the parallelism for internal processing of an element DBMS, for instance parallel processing by *tuple stream*, operations for set, and index operations of temporary relations.

Integrated Environment for Protein Databases

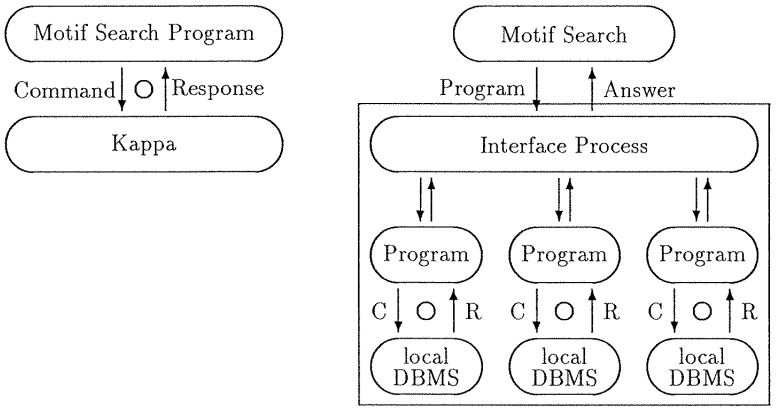
We developed an integrated protein database system on Kappa-P, and a graphic user interface (GUI) for using various protein databases at once.

Purposes

- DBMS suitable for molecular biology
- Visualized and integrated protein database system
- One of the applications for evaluating Kappa-P

Configuration

Publicly distributed protein sequence (PIR), structure (PDB), and feature (ProSite) databases are stored in Kappa-P. A GUI providing an integrated environment for feature representation is employed to permit interactive query and answer through the *remote procedure call* (RPC) between Xwindows and Kappa-P. The *motif search* program is invoked from the GUI and runs in parallel as a filter in each local DBMS of Kappa-P.



Query Processing of Kappa-P

Evaluation

- Suitable Data Model for Existing Public Databases

  The hierarchical data structure of features, for example, is naturally represented in the nested relational model. *Stable* data such as protein names or taxonomy, and *variable* data such as feature descriptions which are often added or corrected by biologists, are stored in separate relations. This is expected to improve processing efficiency.

- Integrated Environment for Protein Databases

  Since existing public protein databases are managed in different institutes, there are various difficulties in using plural databases. For example, feature descriptions of functions are stored in PIR, structural features are in PDB, and relations between amino acid patterns and features are stored in ProSite. The GUI, which is implemented in Xwindows, communicates with Kappa-P via RPC and provides an integrated environment for feature descriptions by showing their positions graphically. It displays functional features in PIR and structural features in PDB, both of which are stored in Kappa-P nested relations.

- Speed-up of Exhaustive Search (Motif Search)

  The most popular use of protein sequence databases is to predict the functions of a function-unknown protein through the homology between its amino acid sequence and those of function-known proteins. *Motif search* is another homology-based search which searches for amino acid patterns within a sequence database. Both require exhaustive searching, and parallel processing is expected to speed up the search process.

  Kappa-P has a feature which passes user-defined programs from the interface process to each local DBMS and executes them. The results are collected by the interface process and returned to the user. Using this mechanism is expected to reduce communication costs considerably, compared to using a mechanism in which whole data are once collected by the interface process and re-distributed to each PE with the user's program.

Demonstration Outline

We performed two kinds of demonstration:

- Kappa-P

  It focuses on the performance and functions of Kappa-P, that is, the effects of horizontally partitioned relations and the introduction of Kappa-P query processing.

- An integrated protein database system on Kappa-P

  It focuses on an application system using protein databases, that is, a GUI for protein features implemented on Xwindows and a protein motif search.

Both demonstrations used protein databases: PIR, PDB, and ProSite. Their specifications (contents, amount [version], and the number of relations in Kappa-P with their names) are as follows:

PIR : Amino acid sequences and characteristic domains of proteins,
60 Mbytes [Sep 1991], 3 relations of sequences, features, and references.

PDB : 3D structure and secondary structures of proteins,
150 Mbytes [Nov 1991], 4 relations of sequences, features, structures, and coordinate matrix.

ProSite : Amino acid sequence patterns (motifs), 508 patterns, 1 Mbytes [May 1991], 1 relation.