5W−7

Integrating Parallel and Distributed Functions into the Persistent Programming Language INADA

Yun JIANG Masayoshi ARITSUGI Guangyi BAI Akifumi MAKINOUCHI
Dept. of Computer Science & Communication Engineering, Kyushu University

1. Introduction

Recently, not only uniprocessor computers but also shared memory multiprocessor computers have been marketed as workstations to be used through a network. This motivates us to implement the persistent objects manipulation in such parallel and distributed environments.

In this paper, we propose a parallel and distributed persistent objects manipulation in the persistent programming language INADA.

2. Objects Manipulation of INADA

INADA is an object-oriented persistent programming language for implementing data-intensive applications[1]. It borrows and extends the object model of C++ so that it provides facilities for handling volatile/persistent objects and for processing queries on collections of these objects.

An object which survives the execution of the procedure which defines it is said to name persistency and is called "persistent object". Persistent objects can be used by other applications over and over again. INADA supports persistent objects by using persistent heaps (PHs).

A PH is a part of a virtual address space and is mapped to a file on a secondary storage. A distributed shared persistent heap server called WAKASHI[2] serves to keep the correspondence between the PH and the file, so that INADA can manipulate persistent objects just like volatile objects.

INADA has three major features:

(1) Persistent heaps and persistent objects: Any object of any class in C++ can be persistent if it is generated on a persistent heap.

(2) Multiple types objects: Persistent objects with multiple types or without any types may exist in INADA.

(3) Set-objects manipulation: INADA provides a data abstraction called set-object for manipulating a large amount of objects. A set-object, either persistent or volatile, can be defined by INADA users.

3. Parallel and Distributed Functions

Based on the basic functions of INADA described above, we add parallel and distributed processing abilities to INADA.

3.2 Parallel Sentence Structure

We introduce INADA parallel sentence structure described as follows.

```
《parallel Execution Sentence》 ::=
parallel do {
  《MethodCall-1》 [| 《MethodCall-i》 ]...
};
《MethodCall-i》 ::=
     [LogicValue=] 《PointerVariable》 ->
        《MethodName》 ([ 《Parameters》 ])
                  ( i=1,2,...n )
```

- A "MethodCall-i" corresponds to a message passing. The object called by the message is either single object or set-object. It may be volatile or persistent. It may be local or remote.
- All "MethodCall-i" sentences enclosed a "parallel do" sentence are simultaneously executed in parallel on multiple threads, and are synchronized[3] when the "parallel do" sentence is finished.
- Any "MethodCall-i" sentence is executed on the site where the object, to which the message is sent, exists. When the object is persistent and on a disk of a remote site, the MethodCall-i becomes a remote procedure call.

- "PointerVariable" contains an object ID. The object ID indirectly contains a site ID where the object exists.
- "Parameters" are used to pass input and output variables for "MethodCall. They can be shared among the methods if these variables are on a shared heap either transient or persistent.
- A "LogicValue" can tell programmers whether the execution of a "MethodCall" is successful or not.

3.1 Approaches

We propose three approaches to implement the parallel sentence structure of INADA.

(1) Parallel Remote Method Execution(PRME)

In order to reduce the cost for manipulating persistent objects, especially persistent set-objects, we propose a way called Parallel Remote Method Execution (PRME) to implement the parallel and distributed processing in IDANA.

It means that INADA allows not only the methods to be executed in parallel on remote machines where the objects exist, but also the parallel executions to happen on the same site where the objects exist.

(2) Virtual Local Multi-thread Environment(VLME)

We introduce the notion called Virtual Local Multi-thread Environment (VLME) so as to implement the parallel distributed persistent objects manipulation easily.

It can be explained in Figure 1. The VLME supports a location transparency for parallel and distributed accesses, because programmers can ignore whether the "MethodCall" is remote or local.

(3) Transient/Persistent Shared Object(T/PSO)

The third feature is called Transient/Persistent Shared Object (T/PSO).

Different from traditional remote method execution, INADA provides transient/persistent objects that are shared by remote/local methods. These objects are allocated on shared heap, either transient or persistent.

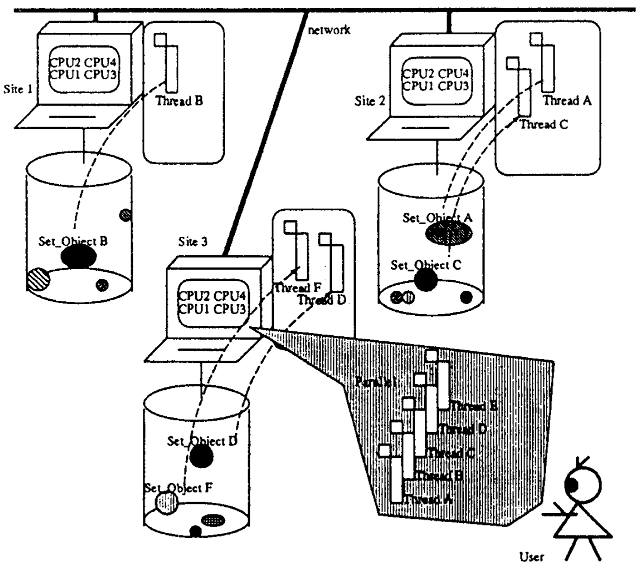

Figure 1. A Virtual Local Multi-Thread Execution Environment

4. Conclusion

We have presented the parallel and distributed method execution that supports the object-oriented programming language INADA.

We also proposed new approaches: PRME, VLME and T/PSO for manipulating of parallel and distributed volatile/persistent objects easily and efficiently.

References

[1] M. Aritsugi and H. Amano, " View in an Object-Oriented Persistent Programming Language," *Proc. of the International Symposium on Next Generation Database Systems and Their Applications*, pp.18-25, September, 1993.

[2]G. Bai and A. Makinouchi, "Implementation and Evaluation of New Approach to Storage Management for Persisted Data -Towards Virtual-Memory Databases," *Proc. of the 2nd Far-East Workshop on Future Database Systems* pp. 211-220, April, 1992.

[3]Y. Jiang and A. Makinouchi, "A Transparent Object-Oriented Synchronization Mechanism for Parallel and distributed Programming," *Proc. of 1993 International Conference on Parallel and Distributed Systems* pp.92-99, December, 1993.