Supporting Ada 95 Passive Partitions in a Distributed Environment

Frank Mueller

Humboldt-Universität zu Berlin, Institut für Informatik, 10099 Berlin (Germany)
e-mail: mueller@informatik.hu-berlin.de phone: (+49) (30) 20181-276

Abstract. Ada 95 passive partitions, containing passive library units, provide the means to distribute data within a network of workstations. This paper shows how passive partitions can be implemented via distributed shared virtual memory (DSM). DSM provides the logical view of a portion of memory shared between physically distributed workstations within a network. In this paper, we relate design issues and operational characteristics of DSM systems to the semantics of shared passive library units as specified in the Ada 95 distributed system annex. We designed and implemented such a DSM system, operating at the granularity of pages, in a portable fashion over POSIX threads. The DSM system establishes its own communication services and is completely independent of any Ada runtime system support. Protected objects are supported via a novel prioritized protocol for distributed mutual exclusion. We integrated the DSM system with the Gnu Ada Translator (Gnat) and its environment to support active partitions. The DSM system integration required minimal changes to the Gnat environment and remains completely transparent to the user and the Ada runtime system. This is the first implementation of shared passive library units within an Ada 95 compilation environment, to our knowledge. The DSM support can provide considerable performance advantages over alternative approaches with repeated remote accesses, since with DSM only the first access to remote data results in communication overhead while consecutive accesses may take the same time as local memory accesses. Using the DSM paradigm, distributed applications can be designed to access distributed shared data efficiently. Furthermore, the shared memory model inherent to the Ada language can be used without modifications in a distributed environment.

1 Introduction

Distributed systems are gaining in popularity because they provide a cost-effective means to exploit the existing processing power of networks of workstations. Nonetheless, there are a number of problems associated with the transition to distributed systems, which often limit the performance gains.

One problem is caused by the communication medium between processing nodes (workstations), which often creates a bottleneck. Recent advances increase the throughput of communications by at least an order of magnitude (e.g., FDDI,
FastEther, ATM), which improves the situation to some extent. However, access to data on remote nodes will always be slower than access to local data. Thus, limiting the number of remote accesses will always be a design goal of distributed systems.

Another problem is rooted in the absence of a global state and the absence of an easily understood programming model to address this issue. As a result, distributed algorithms are often hard to understand and require explicit communication directives (e.g. message passing). Simple solutions often employ a centralized approach, where a single server provides a global state to its clients, but the server may create a bottleneck again.

An alternative solution is provided by distributed virtual shared memory (DSM) [21, 22]. DSM provides the logical view of a portion of an address space, which is shared between physically distributed processing nodes, i.e. a global state. Address references can be distinguished between local memory accesses and DSM accesses, i.e. an architecture with non-uniform memory access (NUMA) is created. However, the well-understood paradigm of concurrent programming can be used without any changes, since the communication between nodes is transparent to the programmer.

The distributed systems annex of Ada 95 [8] supports the abstraction of global data shared between processing nodes via shared passive library units associated with a certain node (called a passive partition). Passive partitions can be supported in various ways. For example, message passing or remote procedure calls (RPC) could be employed to fetch data of passive libraries from a certain node each time it is accessed. A much more efficient alternative is provided by implementing passive partitions via DSM. Here, only the first data access results in communication overhead to obtain ownership of the data. While a node owns the data, consecutive accesses are as fast as local memory accesses. This paper describes how the Ada 95 model of passive partitions can be mapped onto the DSM model.

2 The DSM Model

DSM can be implemented in hardware or in software. This paper focuses on the latter since it is aimed at existing networks of workstations. A major distinction between different DSM systems is given by the granularity of shared data. Sharing can be accomplished at the word level, the page level, at the scope of objects or within tuples. This paper focuses on page sharing because accesses can be monitored at this level via operating system and hardware support available in most systems. Figure 1 depicts such a system with a varying number of CPUs per node (single processor or shared-memory multi-processors) connected by an arbitrary communication medium. A portion of each node's addressing space is reserved for DSM data. While only a subset of all DSM pages may be accessible at a node at a given time (as depicted by the darker shading), this local DSM view is transparent to the user. To the user, the DSM portion of the addressing space seems as a globally consistent, shared data area. For simplicity, it is assumed for now that read and write accesses are not distinguished and that se-