Abstract
Jade is a data-oriented language for exploiting coarse-grain parallelism. A Jade programmer simply augments a serial program with assertions specifying how the program accesses data. The Jade implementation dynamically interprets these assertions, using them to execute the program concurrently while enforcing the program's data dependence constraints. Jade has been implemented as extensions to C, FORTRAN, and C++, and currently runs on the Encore Multimax, Silicon Graphics IRIS 4D/240S, and the Stanford DASH multiprocessors. In this paper, we show how Jade programmers can naturally express hierarchical concurrency patterns by specifying how a program uses hierarchically structured data.

1 Introduction
Jade is a data-oriented language for expressing coarse-grain concurrency. Instead of using explicit control constructs to express the concurrency available in a program, a Jade programmer augments a serial program with Jade constructs that declare how the various sections of the program access data. The Jade implementation uses this information to execute the program concurrently while enforcing the program's underlying data dependence constraints. Jade programmers therefore create coarse-grain parallel programs that preserve the semantics of the original serial programs.

A Jade programmer first divides a sequential program up into tasks. Tasks interact

This research was supported in part by DARPA contract N00014-87-K-0828.
through accesses to *shared objects*. The programmer summarizes each task’s accesses by specifying which shared objects the task will read or write. The Jade implementation uses this information to relax the program’s serial execution order; for example, tasks which access distinct shared objects can execute concurrently. Because each task’s access specification is determined dynamically, Jade programs can exploit data-dependent concurrency available only at run time.

We may contrast Jade’s data-oriented approach to concurrency with the control-oriented approach provided by many parallel programming languages [1, 2, 3, 5]. Control-oriented languages typically provide low-level constructs for creating and synchronizing parallel tasks. These constructs provide precise control over the concurrent behavior of a program. However, it can be difficult to create and maintain parallel programs which contain such a low-level specification of the concurrency structure. Programmers using these low-level constructs must often establish explicit synchronization connections between logically unrelated modules which access the same data. These connections violate the modular structure of the original serial program, making the parallel program harder to understand and modify.

Jade, with its data-oriented constructs, provides a less familiar but conceptually higher-level approach to concurrency. Jade programs only contain local information about the pieces of data that tasks read and write. The Jade implementation, not the programmer, extracts and enforces the global concurrency pattern implicit in the program’s data dependence constraints. By requiring only local data usage information, Jade promotes modularity in parallel programs.

Data-oriented expression of parallelism is not limited to simple concurrency patterns. Simple data structures can be used for simple concurrency patterns such as dynamic task graphs and pipelining [4]. This paper shows that more complicated patterns such as nested levels of parallelism can be achieved via hierarchically structured data. For example, the Jade implementation can generate a parallel tree traversal from a specification of how a program accesses nodes and subtrees. Similarly, a Jade program that accesses a matrix hierarchically as a collection of columns may create a task to perform a matrix operation, which in turn creates tasks to perform the operation on each of the columns in the matrix.

The organization of this paper is as follows. We first briefly review the basic concepts of Jade: the shared objects and the language constructs. We then demonstrate how the use of hierarchical data structures in Jade programs leads to hierarchical concurrency patterns. We also illustrate how the structure of the data hierarchy can be used to refine incrementally the specified accesses of tasks. Finally, we show that hierarchical concurrency makes the generation of concurrency in a Jade program more efficient.