Abstract
A number of programming language models, including actors, provide inherent concurrency. We are developing high-level language constructs using actors and studying their implementation on multiprocessor architectures. This report describes our experience with programming in actors by means of a specific example of scientific computation. We also discuss work in progress on compilation technology for efficient program execution on multiprocessors.

1 Introduction
Concurrent language models, such as concurrent logic programming, functional programming and actors, provide inherent concurrency in the evaluation of expressions. However, unlike other models, actors allow state to be directly expressed and manipulated. Our experience suggests that this enables us to write programs which not only avoid unnecessary
sequencing of actions but which are also easily understandable. We discuss the structure of actor languages and issues related to their implementation on distributed memory architectures.

Inherently concurrent languages often suffer from an embarrassing amount of concurrency. Current compiler technology is not sufficiently developed to optimize placement and migration of objects in order to provide sufficient execution efficiency on distributed memory architectures. Furthermore, in some cases, the structure of a problem is crystalline and well-understood by the programmer. In these cases, the use of annotations for specifying processor locations for expression evaluation to guide the runtime system has been suggested (e.g. [11]). However, in the context of actors, explicit message passing primitives imply that a separate syntax to specify such annotations is superfluous.

The organization of this paper is as follows. In Section 2 we give a brief overview of the actor model. Section 3 outlines multi-send constructs to increase concurrency. Section 4 illustrates concurrency in actors by an algorithm namely the Cholesky Decomposition of an SPD matrix. In Section 5 we discuss the representation of actors on multiprocessors. The final section presents some conclusions.

2 The Actor Model

Actors are self-contained, interactive, independent components of a computing system that communicate by asynchronous message passing. Each actor has a conceptual location, its mail address, and a behavior. An actor's acquaintances are all of the actors whose mail addresses it knows. In order to abstract over processor speeds and allow adaptive routing, preservation of message order is not guaranteed. However, messages sent are guaranteed to be received with an unbounded but finite delay.

State change in actors is specified using replacement behaviors. Each time an actor processes a communication, it also computes its behavior in response to the next communication it may process. The replacement behavior for a purely functional actor is identical to the original behavior; in general it may change. The change in the behavior of an actor may represent a simple change of state variables, such as change in the balance of an account, or it may represent changes in the operations (methods) which are carried out in